Real-time Viscous Thin Films

ORESTIS VANTZOS, Technion–IIT, Mathematics Department
SAAR RAZ, Technion–IIT, Computer Science Department
MIRELA BEN-CHEN, Technion–IIT, Computer Science Department

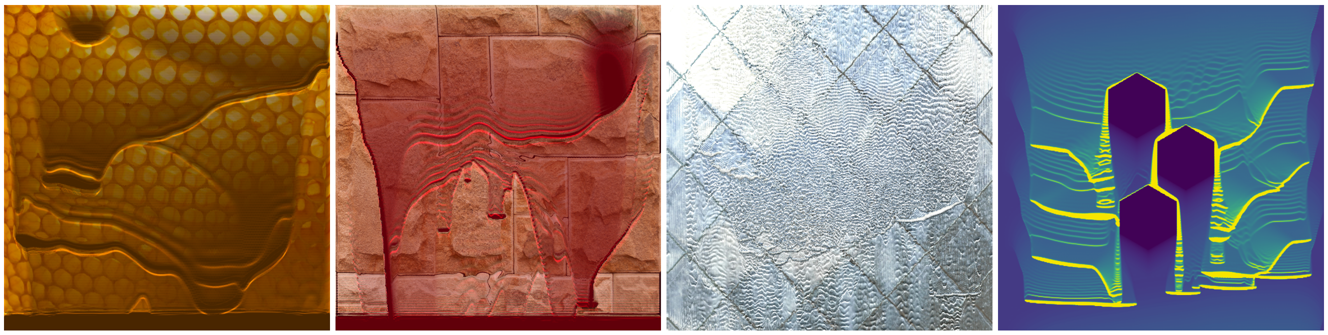


Fig. 1. Images captured during real-time simulation of viscous fluids.

We propose a novel discrete scheme for simulating viscous thin films at real-time frame rates. Our scheme is based on a new formulation of the gradient flow approach, that leads to a discretization based on *local stencils* that are easily computable on the GPU. Our approach has physical fidelity, as the total mass is guaranteed to be preserved, an appropriate discrete energy is controlled, and the film height is guaranteed to be non-negative at all times. In addition, and unlike all existing methods for thin films simulation, it is fast enough to allow realtime interaction with the flow, for designing initial conditions and controlling the forces during the simulation.

CCS Concepts: • **Computing methodologies → Computer graphics**; **Physical simulation**; **Parallel algorithms**; • **Applied computing → Physics**;

Additional Key Words and Phrases: ACM proceedings, Thin Films, Fluid Dynamics, Interactive Simulation, GPU Computing.

ACM Reference Format:
Orestis Vantzos, Saar Raz, and Mirela Ben-Chen. 2018. Real-time Viscous Thin Films. *ACM Trans. Graph.* 37, 6, Article 281 (November 2018), 10 pages. https://doi.org/10.1145/3272127.3275086

1 INTRODUCTION

The intricate behavior of viscous thin films has fascinated physicists, mathematicians and engineers for many years [Craster and Matar 2009; Oron et al. 1997]. With the advent of mobile devices with graphics hardware, it is a natural question whether such liquids can be simulated in real time and controlled by the user, using the mobile display as the substrate layer on top of which the liquid flows.

The physics driving the evolution of the thin film is governed by a fourth order partial differential equation, whose existing numerical evolution schemes are not computationally efficient enough to run at interactive rates [Vantzos et al. 2017]. Schemes that can run at interactive rates [Goswami et al. 2010], cannot simulate highly viscous fluids and their attendant intricate behavior.

We propose a novel numerical scheme for simulating the thin film equation on a planar domain, with gravity and other forces. Our scheme is based on a modification of the *lubrication approximation* [Oron et al. 1997], where the fluid is represented as a height function over a planar domain. Our modification adds a quadratic term to the governing equation, that stabilizes the flow while preserving the visual fidelity of the simulation. Our time and space discretization is based on the *gradient flow* approach [Otto 2001], and guarantees exact conservation of mass, and non-negativity of the height function. Finally, the numerical scheme is *local*, and thus easily parallelizable on the GPU, without requiring costly memory access. We implemented the approach using WebGL, and demonstrate that it can run at interactive rates on mobile devices, allowing the user to interact with the flow by adding liquid, obstacles, and control the direction of gravity.

1.1 Related Work

Fluid simulation is a massive topic and the recent book by Bridson [2015] can serve as an excellent introduction. We are interested in a specific regime of fluids, namely viscous thin films flowing due to gravity and other external forces. As such, generic fluid solvers that are based on simulating the full Navier-Stokes equations are unnecessarily complex for this task. We thus focus on the simulation of fluids in this regime, with specialized tools.

The work was supported by the European Research Council (ERC starting grant No. 714776 "OPREP"). The background images were obtained from www.pixabay.com and www.pexels.com under a Creative Commons License.
Authors' addresses: Orestis Vantzos, Technion–IIT, Mathematics Department, Haifa, Israel, ovantzos@technion.ac.il; Saar Raz, Technion–IIT, Computer Science Department, Haifa, Israel, saar@raz.email; Mirela Ben-Chen, Technion–IIT, Computer Science Department, Haifa, Israel, mirelab@cs.technion.ac.il.

0730-0301/2018/11-ART281
https://doi.org/10.1145/3272127.3275086

Viscous thin films in Physics. The governing equations of viscous thin films have been researched for many years, both experimentally and numerically. Oron et al. cover in detail the earlier work [1997], and Craster et al. provide a more recent review [2009]. The behavior of the contact line between the fluid and the dewetted region has also generated much research interest [Snoeijer and Andreotti 2013], as has the behavior of a thin layer on an inclined plane [Kalliadasis et al. 2011].

From a numerical perspective, thin films are often simulated using the *lubrication approximation*, which is a reduced Navier-Stokes model based on the assumption of the small thickness of the film. This is a fourth order PDE, leading to time step size difficulties for explicit schemes (see e.g. [Grün and Rumpf 2000; Zhornitskaya and Bertozzi 1999]). Alternatively, a variational formulation can be derived, by considering the film evolution as a *gradient flow*, on an abstract Riemannian manifold, where the Riemannian metric encodes the resistance of the film to move due to its viscosity (see e.g. [Otto 2001]). Such schemes have a natural time discretization that preserves the structure of the flow, such as its total mass, and is stable and numerically robust. Our approach is based on a small modification of the lubrication approximation, which is not meant to be physically accurate, yet has a stabilizing effect on the flow while remaining visually plausible. Furthermore, we provide a flux-based gradient flow formulation that, in contrast with existing approaches, leads to a completely *local* numerical scheme.

Viscous thin films in Computer Graphics. In the Graphics community viscous fluids have been simulated as free surface flows, from the pioneering work of Carlson et al. [2002], to the most recent treatment by Larionov et al. [2017], that also includes an excellent review of the topic. However, these methods do a full scale three dimensional simulation, and do not take advantage of the lower dimensional properties of thin films. Lagrangian methods that leverage the reduced dimension of viscous threads [Bergou et al. 2010] and sheets [Batty et al. 2012] and Lagrangian co-dimensional methods that represent the fluid using a simplicial complex [Zhu et al. 2015, 2014] have better computational complexity, but still do not run at interactive rates. Real time techniques for simulating the shallow water equations can generate intricate wave effects (see [Wang et al. 2007] and references within), however they are not appropriate for very high viscosity liquids, such as honey.

Closer to our approach, Eulerian gradient flow formulations for thin films on curved surfaces have been proposed, using flux-based [Rumpf and Vantzos 2013] and velocity-based [Vantzos et al. 2017] approaches. These lead to a sparse linear solve per iteration, and are therefore not amenable to real-time computation on highly resolved meshes. Furthermore, these schemes cannot guarantee the non-negativity of the height field during the simulation, leading to potential instabilities. An interactive simulation of the Hele-Shaw flow, that can be seen as a special case of a thin film flow, was proposed by Segall and co-authors [2016]. That approach simulates the contact curve of the fluid using complex-valued functions and conformal maps, does not incorporate gravity, and cannot handle more than one connected component of fluid.

Interactive fluid simulation. Fluid simulation is traditionally a heavy computational task, leading to various challenges for controlling the initial conditions and the forces during the simulation, especially in Computer Graphics applications where the fluids should be easily directable by an artist. In recent years, GPUs have become an important tool for more efficient computations in many applications, including in fluid simulation [Goswami et al. 2010; Harris 2005; Navarro-Hinojosa et al. 2018]. Furthermore, such interactive fluid simulations can now run on mobile devices [Harwood and Revell 2018], enabling the user to control the simulation in real-time, for instance via the touch interface, [Chen et al. 2015; Stuyck et al. 2017]. Nevertheless, and to the best of our knowledge, there do not currently exist real time fluid simulators that are capable of generating the viscous thin film effects that we demonstrate. Even more so, as our approach is based on a variational model, and thus guarantees fluid mass preservation and the reduction of a discrete energy.

1.2 Contributions

Our main contribution is a numerical scheme for viscous thin film simulation that

- is defined via local operations, and is thus highly efficient and easy to implement as a shader,
- is derived using a local gradient flow formulation that guarantees important theoretical properties, namely mass preservation, non-negativity of the solution and control of a suitable discrete energy,
- it can be implemented on mobile devices with responsive user interaction via accelerometer (to change the direction of gravity) or a touch interface (to add fluid or place obstacles).

2 METHOD

2.1 Physics

Consider the classic thin film equation [Oron et al. 1997]

$$\frac{\partial u}{\partial t} = \operatorname{div}(M(u)\nabla w), \quad M(u) = \frac{u^3}{3}, \quad w = W - \epsilon\Delta u \tag{1}$$

which describes the evolution of the *mass-per-surface-area* $u(x,t)$ of a viscous liquid thin film of *typical thickness* ϵ (so that the local thickness of the film is $\sim \epsilon u$), driven by variations in the *potential* w due to the influence of *surface tension* (the $\epsilon\Delta u$ term) and external forces such as gravity (the *external potential* $W(x,t)$). The motion of the fluid is also non-linearly dependent on the local mass concentration through the *mobility* $M(u)$, which reflects the retarding effect that viscous friction inside the fluid has on the flow of the film. For the rest of the paper, it is useful to rewrite (a slightly modified version of) equation (1) in terms of the *flux* f:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \operatorname{div} f = 0, \\ &f = -M(u)\nabla(W - \epsilon\Delta u + \eta u), \\ &u \geq 0 \end{aligned} \tag{2}$$

The extra term is a stabilising anisotropic second-order (as opposed to the 4th-order diffusion $\epsilon\Delta u$ term) diffusion term, that is quite useful in practice. We have also made explicit the, physically necessary, condition that the density u can not be negative.

The thin film equation (1) has the following properties: a) it preserves the *total mass* $\int u \, dx$ and b) the *Dirichlet energy* $\frac{1}{2}\int|\nabla u|^2 \, dx$ (which is a measure of the smoothness of the solution) is controlled; in particular, in the absence of external potential $W = 0$ it is non-increasing. Regarding the non-negativity of the solutions of the thin film equation, there has been a lot of theoretical [Bertozzi et al. 2001] and numerical [Grün and Rumpf 2000] work, and it has indeed proven to be quite a challenge from the computational point of view. It is important to preserve these properties in the discrete setting, as they play an important role in both the numerical stability and physical fidelity of the simulation.

2.2 Time Discretization with Gradient Flows

One approach to deriving discrete schemes for a wide range of evolution equations, that include the thin film equation (1), is to take advantage of their *gradient flow* structure [Otto 2001], i.e. the fact that they can be seen in a certain sense as a steepest gradient descent for a suitable energy functional. This point of view leads to variational discrete schemes of the *minimizing movement* type [Giorgi and Ambrosio 2006], where a (constrained) minimization problem of the (abstract) form

$$u^{n+1} = \underset{u\in\mathcal{R}(u^n)\subset X}{\operatorname{argmin}} \left\{ \frac{1}{2\tau}\operatorname{dist}^2_X(u,u^n) + \mathcal{E}(u) \right\} \tag{3}$$

needs to be solved at each time step. This equation is to be understood as follows: the (approximate) solution $u^{n+1} \in X$ at time $t^{n+1} = t^n + \tau$, where X is a suitable space, is the minimizer of a combination of the free energy $\mathcal{E} : X \to \mathbb{R}$ and the distance (in X) from u^n, over a subset $\mathcal{R}(u^n) \subset X$ of states that are "reachable" from u^n. For thin film-type equations, the set of reachable states is associated with an evolution law of the form $\frac{\partial u}{\partial t} + \mathcal{D}_u\phi = 0$, where $\mathcal{D}_u$ is a differential operator and $\phi \in Y$ is an auxiliary variable (in a separate space Y), so that $u \in \mathcal{R}(u^n)$ iff there exists $\phi \in Y$ so that $u - u^n + \tau\,\mathcal{D}_{u^n}\phi = 0$. We can use this connection to rewrite the distance between u^n and any (reachable) u as a function of the ϕ that takes us from u^n to u, leading to schemes of the form:

$$(u^{n+1},\phi^{n+1}) = \underset{\substack{u\in X,\phi\in Y\\ u-u^n+\tau\,\mathcal{D}_{u^n}\phi=0}}{\operatorname{argmin}} \left\{ \frac{\tau}{2} g_{u^n}(\phi,\phi) + \mathcal{E}(u) \right\} \tag{4}$$

In the language of *differential geometry*, one can think of X as a manifold and Y as the tangent space around u^n, with the equation $u - u^n + \tau\,\mathcal{D}_{u^n}\phi = 0$ serving as the exponential map that maps tangent vectors ϕ to nearby points u, so that the distance $\operatorname{dist}_X(u,u^n)$ is naturally connected to the metric $g_{u^n}(\phi,\phi)$ at u^n. As stated before then, this can indeed be seen as an attempt to flow in the direction of steepest descent of $\mathcal{E}$, but in a generalized manifold setting.

There are various ways to apply the abstract framework described above to the problem (2); one can choose the *flux* f as the auxiliary variable, so that $u - u^n + \tau \operatorname{div} f = 0$ and $g_{u^n}(f,f) = \int M(u^n)^{-1}|f|^2 \, dx$, or use a *velocity* v instead so that $u - u^n + \tau \operatorname{div}(uv) = 0$ with a suitably modified metric. Both approaches have been used, the flux-based in [Rumpf and Vantzos 2013] and the velocity-based in [Vantzos et al. 2017] for instance. Our scheme is based on the flux formulation of the gradient flow for a suitable energy $\mathcal{E}(u)$:

$$\begin{aligned}
(u^{n+1},f^{n+1}) &= \underset{(u,f)\in\mathcal{R}(u^n)}{\operatorname{argmin}} \left\{ \frac{\tau}{2}\int M(u^n)^{-1}|f|^2 \, dx + \mathcal{E}(u) \right\} \\
\mathcal{E}(u) &= \int \frac{\epsilon}{2}|\nabla u|^2 + W(x)u + \frac{\eta}{2}|u|^2 \, dx \\
\mathcal{R}(u^n) &= \{(u,f) \mid u - u^n + \tau \operatorname{div} f = 0,\ u \geq 0\}
\end{aligned} \tag{5}$$

Schemes that are based on direct discretizations of this type of constrained optimization problem have important advantages, such as guaranteed mass conservation and energy reduction, and consequently unconditional stability. On the other hand, especially in the context of real-time GPU-based simulation, they also suffer from certain disadvantages, namely the need for inverting large sparse matrices at each time-step and the question of how to represent vector based quantities, such as the flux and the velocity, in a GPU-friendly format.

We work around these issues by combining the gradient-flow approach with the *fractional step method*. We assume that the flux f is a linear combination of a number of fixed locally-supported fluxes f_i, i.e. $f = \sum_{k=1}^{K}\lambda_k f_k$. According to the fractional step method, the effect of applying the flux f to u for a time interval τ, which we can denote in operator form as $T_{\tau f}u$, can be approximated by applying the partial fluxes f_k sequentially: $T_{\tau f} \approx T_{\tau\lambda_k f_k} \dots T_{\tau\lambda_1 f_1}$. A single time-step of the scheme can then be written as

$$\begin{aligned}
u^{(0)} &= u^n \\
u^{(k)} &= u^{(k-1)} - \tau\lambda(u^{(k-1)}, f_k) \operatorname{div} f_k \\
u^{n+1} &= u^{(K)}
\end{aligned} \tag{6}$$

The gradient flow scheme (5) can be used then to determine the magnitude of each (predetermined) partial flux individually.

2.3 Discrete Local Fluxes

To derive a fully discrete scheme for the problem (2), we consider a uniform Cartesian grid of $N_G \times M_G$ cells, with uniform size h in both dimensions, and periodic boundary conditions. The discrete fluid density $u_h \in \mathbb{R}^{N_G\times M_G}$ is represented by a rectangular array with values u_{ij}, and likewise for the discrete potential W_h, whose entries are $W_{ij} = W(x_{ij})$, x_{ij} being the center of the (i,j) cell. In the spirit of the *finite volumes* method, we discretize the fluxes over the edges $p \to q$ between neighboring cells $p = (i,j)$ and $q = (i',j')$, representing flow in the i-direction $f_{(i,j)\to(i+1,j)}$ or in the j-direction $f_{(i,j)\to(i,j+1)}$.

Our scheme is designed to reduce the following discrete energy:

$$\mathcal{E}_h(u_h) = \frac{\epsilon}{2h^2}\sum_{p\to q}|u_p - u_q|^2 + \sum_p W_p u_p + \frac{\eta}{2}\sum_p |u_p|^2 \tag{7}$$

The first term is a discrete Dirichlet energy, and is a measure of how smooth the solution is. Controlling it is therefore important for stability. The second term drives the movement of the fluid to areas of low external potential W, for instance from high altitude to low altitude for gravity. Finally, the last term penalizes high concentrations of mass and also acts as a slope limiter; it provides extra stabilisation and improves the visual effect, especially in the presence of very high gradients of W or large time-steps.

Following the minimizing movements time discretization (5), we minimize the sum of a suitable norm of the flux f and the discrete energy (7). As per the fractional step scheme (6), instead of performing this minimization for the flux field over the entire domain, we do it locally for each flux $f = f_{p\to q}$ between two adjacent cells p and q. Using $\tilde{u}$ to denote the updated densities after the flow, and writing only the relevant terms of the energy in the immediate neighbourhood of the cells, we get the following:

$$\min_{f\in\mathbb{R}}\left\{\frac{\tau|f|^2}{2\,M(u_p,u_q)}+\frac{\epsilon}{2h^2}\sum_{p'\to q'}|\tilde{u}_{p'}-\tilde{u}_{q'}|^2 + (W_p\tilde{u}_p+W_q\tilde{u}_q)+\frac{\eta}{2}(|\tilde{u}_p|^2+|\tilde{u}_q|^2)\right\}$$

$$\tilde{u}_p = u_p-\frac{\tau}{h}f,\quad \tilde{u}_p\geq 0$$

$$\tilde{u}_q = u_q+\frac{\tau}{h}f,\quad \tilde{u}_q\geq 0$$

The sum in the middle term is over all the edges between p and q and their neighbours (and each other) (see Fig. 2).

Plugging the updated densities into the objective function, we eventually get a constrained quadratic optimization problem of the form:

$$\min_{a\leq f\leq b}\left\{\frac{1}{2}\alpha f^2-\beta f+\gamma\right\}$$

with

$$\alpha=\frac{\tau}{M(u_p,u_q)}+\frac{2(5\epsilon+\eta h^2)\tau^2}{h^4}$$

$$\beta=\frac{\tau}{h}\left\{\epsilon((\Delta_h u)_q-(\Delta_h u)_p)+(W_q-W_p)+\eta(u_q-u_p)\right\}$$

(γ is not needed)

$$a=-\frac{h}{\tau}u_q,\quad b=\frac{h}{\tau}u_p$$

with the discrete 5-point Laplacian $\Delta_h = \frac{1}{h^2}\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix}$. The non-negativity of the discrete mobility ensures that $\alpha>0$ and therefore the quadratic indeed has the unique minimum $f=\frac{\beta}{\alpha}$. Finally, recall that when the global minimum of a (convex) quadratic function is outside of the range $[a,b]$, then the minimum over the range is simply whichever of the bounds of the interval is closest.

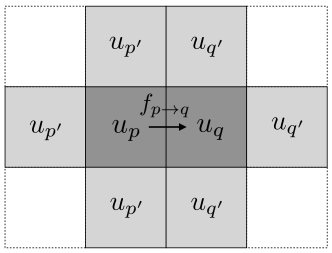


Fig. 2. Flux $f_{p\to q}$ between two adjacent cells p and q. The flux depends on the values in the (5-cell) neighborhood of the two cells. See also Fig. 4a.

This leads to the following *numerical flux* $f_{p\to q}$ between two neighbouring cells:

$$f=-\frac{M(u_p,u_q)}{\theta\, h}\Big\{W_q-W_p-\epsilon\left((\Delta_h u)_q-(\Delta_h u)_p\right)+\eta\,(u_q-u_p)\Big\}$$
$$f_{p\to q}=\max(-\tfrac{h}{\tau}u_q,\min(f,\tfrac{h}{\tau}u_p)) \tag{8}$$

with the following action, over a time interval of duration τ, on the density of the cells:

$$\begin{aligned}u_p &\leftarrow u_p-\tfrac{\tau}{h}f_{p\to q}\\ u_q &\leftarrow u_q-\tfrac{\tau}{h}f_{q\to p}=u_q+\tfrac{\tau}{h}f_{p\to q}\end{aligned} \tag{9}$$

This is a rather straightforward discretization of (2), except for the clamping of the flux (to ensure non-negativity) and the regularizing parameter $\theta := 1+\frac{2\tau M(u_p,u_q)(5\epsilon+\eta h^2)}{h^4}$, which is necessary for the proper energetic behaviour of the scheme.

The *discrete mobility* $M(\cdot,\cdot)$ can be defined in many ways, as long as it is symmetric, $M(u_1,u_2)>0$ for any positive $u_1\neq u_2$, and $M(u,u)=\frac{u^3}{3}$. There are good theoretical arguments [Grün and Rumpf 2000] in favour of the discrete mobility $M(u_1,u_2)=\frac{2u_1^2u_2^2}{3(u_1+u_2)}$, but we have explored other options too, such as $M(u_1,u_2)=\frac{2}{3}(u_1^{-3}+u_2^{-3})^{-1}$ (see Fig. 3).

The local update (8)-(9) has the following important properties:

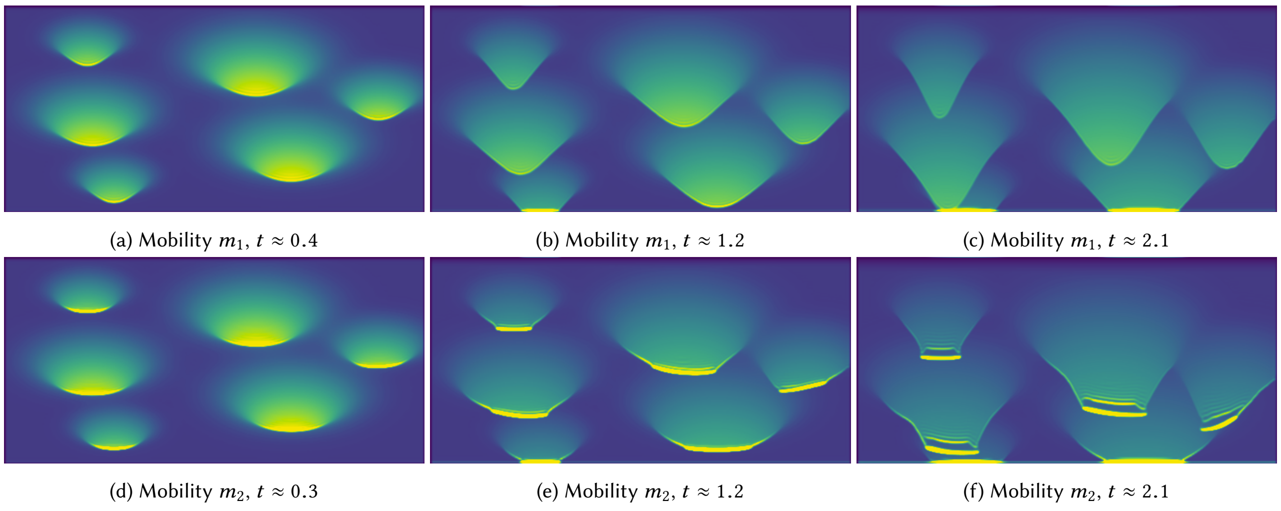
- *Mass preservation:* Since the same amount of fluid is removed from one cell and added to the other, the total mass $\sum_p u_p$ always remains constant.
- *Non-negativity:* Follows immediately from the min-maxing operation in (8). It is important to note that, given $u_p,u_q\geq 0$ before the update, $-\frac{h}{\tau}u_q\leq 0\leq\frac{h}{\tau}u_p$ and $f_{p\to q}$ is well-defined.
- *Energy reduction:* By construction, the flux $f_{p\to q}$ and the corresponding updated density $\tilde{u}_h$ minimize the sum $\frac{\tau|f|^2}{2\,M(u_p,u_q)}+\mathcal{E}_h(\tilde{u}_h)$, over any other flux $f'\in[-\frac{h}{\tau}u_q,\frac{h}{\tau}u_p]$ and its associated density $\tilde{u}'_h$. The key observation is that the null flux $f'=0$, which corresponds to the non-updated density u, is indeed within that range, and so the update does not increase the energy:

$$\frac{\tau|f|^2}{2\,M(u_p,u_q)}+\mathcal{E}_h(\tilde{u}_h)\leq 0+\mathcal{E}_h(u_h)$$
$$\Rightarrow\mathcal{E}_h(\tilde{u}_h)\leq\mathcal{E}_h(u_h).$$

2.4 Fully Discrete Parallel Scheme

To fully utilize the GPU's parallel computing power, we apply the local updates to sets of edges in parallel. To avoid race conditions we must first break the set of edges into *passes*, where each pass contains edges whose updates do not depend on cells adjacent to other edges in the pass. This induces a *domino* relaxation pattern at each pass, as illustrated in Fig. 4a. The horizontal edges are divided into 4 sets, and likewise for the vertical, for a total of 8 passes. Fig. 4b shows how the passes cover the entire set of edges. See algorithm 1 for a pseudocode of this scheme.

To coordinate the threads that act on the various cells in parallel, each pass is identified via a *direction vector* (d_i,d_j), which is $(1,0)$



(a) Mobility m_1, $t \approx 0.4$ (b) Mobility m_1, $t \approx 1.2$ (c) Mobility m_1, $t \approx 2.1$

(d) Mobility m_2, $t \approx 0.3$ (e) Mobility m_2, $t \approx 1.2$ (f) Mobility m_2, $t \approx 2.1$

Fig. 3. Flow for different discrete mobilities, $m_1(u_1, u_2) = \frac{2}{3}(u_1^{-3} + u_2^{-3})^{-1}$ and $m_2(u_1, u_2) = \frac{2u_1^2 u_2^2}{3(u_1+u_2)}$. Although both mobilities approximate the same continuous mobility $M(u) = \frac{1}{3}u^{-3}$, they result in different flow rates between empty and full cells, and so produce advancing droplets with distinct shapes.

for horizontal and $(0, 1)$ for vertical passes, and a *parity index* $\rho \in \{0, 1, 2, 3\}$. From these we calculate for each cell (i, j) the *parity*

$$\rho_{ij} = ((d_j + 1)i + (d_i + 1)j + \rho) \bmod 4 \ \in \{0, 1, 2, 3\},$$

which is illustrated in Fig. 4a. During a given pass, cells with parity 0 and 1 are paired, and likewise for cells with parity 2 and 3; cells with even parity are paired with cells in the direction (d_i, d_j), whereas cells with odd parity are paired with cells in the opposite direction $(-d_i, -d_j)$ (lines 4-9 in alg. 1). This ensures that both cells in a pair calculate the same flux in magnitude, but with opposite signs (since ij and $i'j'$ are effectively exchanged in line 14 of the alg.). Finally, only the cell pairs with parity 2 or 3 are allowed to update their values (see again Fig. 4a). As the parity index ρ goes from 0 to 3, the parity of each cell also changes, giving it the opportunity to exchange mass with each of its neighbors (when its parity is 2 or 3). The process is then repeated in the other direction by flipping the direction vector (d_i, d_j). The careful partitioning of the local edge updates into passes ensures that the global scheme has the same properties that we proved for the local scheme, i.e. it is also *mass and non-negativity preserving* and *energy reducing*.

3 IMPLEMENTATION

3.1 WebGL Implementation

We based our implementation on WebGL [Jackson and Gilbert 2018], a browser-based version of the OpenGL API; it allowed us to use the same code on various devices, from mobile phones to desktop computers with powerful dedicated GPUs. Moreover, this gave us native access to the touch screen and orientation hardware on mobile devices via the built-in Javascript API.

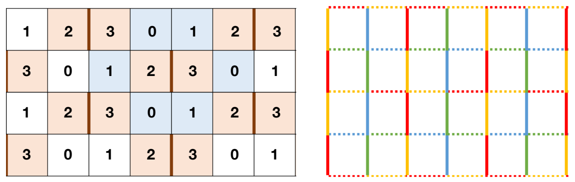
Our numerical scheme is implemented as a *fragment shader*, and the fluid density u is stored as a *floating point texture*. We maintain two such textures, using one as input to the shader and rendering to the other, switching between the two for each pass (*double buffering*).

The external potential W is made available to the shader as an additional texture, whereas the various parameters are passed as uniforms. After the final simulation pass, the fluid density texture is passed through a visualisation fragment shader and rendered to the screen.

3.2 Dynamic Time Stepping

Although from a theoretical point of view the speed of flow of the fluid is regulated by the time step parameter τ, in the real-time setting the perceived speed of flow also depends on the number of scheme iterations, i.e. performing more iterations per frame makes the fluid appear to advance at a faster rate. It follows that the actual frame rate also affects the apparent speed of the simulation, meaning devices with more GPU power would display faster simulations. To compensate for these factors, we determine the time step at each frame (denoted τ) dynamically.

The basic idea behind the dynamic time stepping is that each full (visiting edges of all parities and directions) relaxation pass of the algorithm does a certain amount of work towards propagating the



(a) Cell parities. Updated cells in orange, cells being read in blue. Note that two edges in the same pass may read (but not write) the same cell.

(b) Red, blue, green and yellow mark the 8 different passes (4 vertical, 4 horizontal). Horizontal edges are marked with dotted lines.

Fig. 4. Partitioning of edges into passes

fluid proportional to the time step, i.e. $\tau \sim$ *work/pass.* A uniform movement rate then corresponds to a fixed total amount of work per second:

$$\tau \cdot \#_{\text{passes/frame}} \cdot \#_{\text{frames/sec}} \approx \text{work/sec} = const \tag{10}$$

We generally strive to have an adequate number of passes/frame while maintaining a high framerate, leaving τ as the free parameter that is used to maintain a uniform rate of motion. See Fig. 5 for an illustration of the effect that different time step and pass/frame combinations have on the simulation. Due to hardware constraints, one might end up affording only a small number of passes/frame while maintaining a minimum framerate; one can then attempt to 'drive' the fluid harder by increasing the external potential W. It is in this case where increasing the diffusion (dampening) parameter η is particularly helpful (Fig. 6).

3.3 Driving the Flow via the External Potential

As the fluid tends to flow against the gradient of the external potential, i.e. from areas of high W to areas of low W, the external potential can be used to drive the fluid around the domain. The basic setting is a simple linear gradient, which corresponds to constant gravity. Its effect can be seen in all the figures in the paper, and the accompanying video. On mobile phones we can even dynamically align the direction and strength of the gradient based on the orientation of the device.

Another factor that can be added to W is the geometry of the underlying surface. The physically proper way to do this is by subtracting the *mean curvature* H of the underlying surface from W, as surface tension causes the fluid to concentrate in areas of positive mean curvature such as grooves or holes. The full interaction between the thin film and the geometry is quite complicated [Vantzos et al. 2017], but this first-order approximation is adequate in this context. In fact, if the relief of the underlying surface is available as a height-map R, one can get visually convincing results by simply taking $W \leftarrow W + \lambda R$, so that the fluid tends to accumulate in areas of low relief (such as cracks). See Fig. 8b for an example of this.

3.4 Other Parameters

Apart from the time step τ, the behaviour of the scheme is also influenced by the other parameters. Stronger gradients in the external potential W (such as a very steep gravity gradient) make for a faster motion of the fluid, but they also act in a (physically) destabilising manner, so that oscillations can appear. This is particularly true when the time step is large and/or the number of iterations per frame is low. The parameters ϵ and η are both stabilising, and can therefore be increased to counter the aforementioned instability. The parameter ϵ serves as a typical length scale for the various features of the viscous flow, such as droplets or fronts. The diffusion parameter η on the other hand stabilises the flow (Fig. 6) by penalising large concentrations of fluid. Intuitively, both parameters make the fluid appear more viscous ("thick"), although they are not equivalent; Fig. 7 illustrates the effect of these parameters.

3.5 Rendering

Our algorithm outputs a *height map* representing the fluid mass at each pixel. In our demonstration application we implemented a basic refraction shader with normals calculated from the height map, along with illumination based on *caustics* using the algorithm presented by [Yuksel and Keyser 2009].

3.6 Boundary Conditions

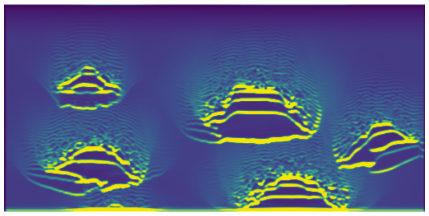
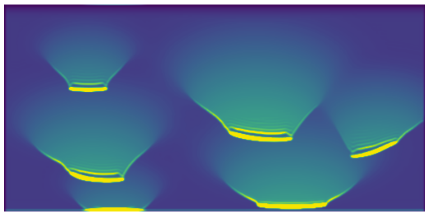
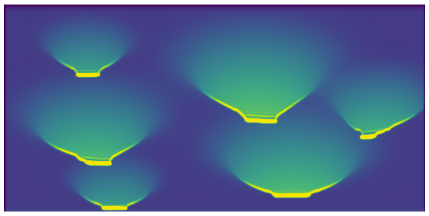
By using OpenGL we get automatic support for *periodic boundary conditions* by using textures with 'repeat' configurations. *Neumann boundary conditions*, which are also useful in many applications, can be implemented by enforcing zero flux across the boundary edges. Note that for the discrete mobility functions that we use, $u_1 = 0$ or $u_2 = 0 \Rightarrow M(u_1, u_2) = 0$, meaning that no mass can flow in/out of pixels where $u = 0$ (*dewetting*). We use this fact to implement the desired boundary conditions by simply setting the values of u of pixels on the boundary to 0 explicitly. We also take advantage of the dewetting effect to let the user draw obstacles interactively (see sec. 4.3).

3.7 Limitations

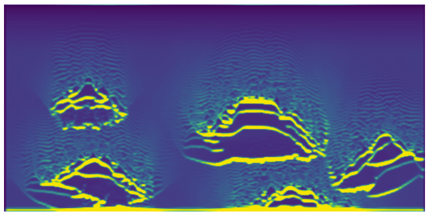
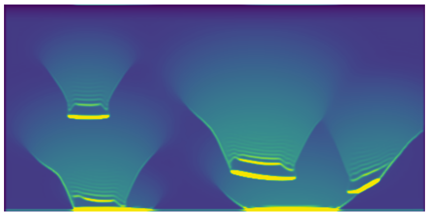
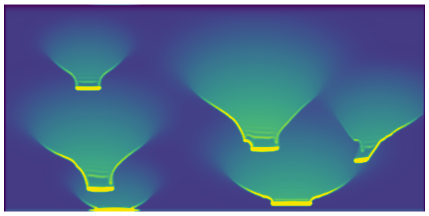
One limitation intrinsic to the local nature of the updates of the scheme, is that information can only travel a limited number of cells per iteration. Methods that involve a non-local step, such as inverting a matrix for instance, do not suffer from this as every cell is potentially coupled to every other cell within a single time-step. At low iteration-per-frame counts this can artificially limit the effective flow rate of the fluid. In practice most visually interesting features of the flow, such as droplets, are local in nature and the scheme allows for adequate iterations per frame even on low powered devices such as mobile phones.

ALGORITHM 1: Parallel update scheme

```
input : Current fluid density $u$,
        parameters $(h, \tau, \epsilon, \eta)$, external potential $W$
output: Updated $u$ with $u \geq 0$.
foreach edge direction $(d_i, d_j) \in (1,0), (0,1)$ do
    foreach parity $\rho \in \{0,1,2,3\}$ do
        parallel foreach thread assigned to pixel $(i, j)$ do
            $\rho_{ij} \leftarrow ((d_j + 1)i + (d_i + 1)j + \rho) \bmod 4$
            if $\rho_{ij} \in \{0, 2\}$ then
                $(i', j') \leftarrow (i + d_i, j + d_j)$
            else
                $(i', j') \leftarrow (i - d_i, j - d_j)$
            end
            $\Delta_{ij} \leftarrow (-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})/h^2$
            $\Delta_{i'j'} \leftarrow (-4u_{i'j'} + u_{i'+1,j'} + u_{i'-1,j'} + u_{i',j'+1} + u_{i',j'-1})/h^2$
            $m \leftarrow M(u_{ij}, u_{i'j'})$
            $\theta \leftarrow 1 + 2\tau m(5\epsilon + \eta h^2)/h^4$
            $f \leftarrow -\frac{m}{\theta h}\left((W_{i'j'} - W_{ij}) - \epsilon(\Delta_{i'j'} - \Delta_{ij}) + \eta(u_{i'j'} - u_{ij})\right)$
            $\delta u \leftarrow \max(-u_{i'j'}, \min(\frac{\tau}{h}f, u_{ij}))$
            if $\rho_{ij} \in \{2, 3\}$ then
                $u_{ij} \leftarrow u_{ij} - \delta u$
            end
        end
    end
end
```

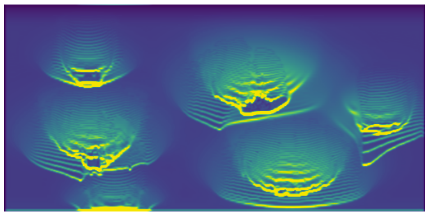
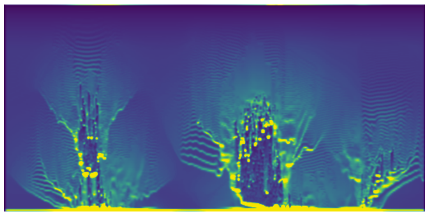
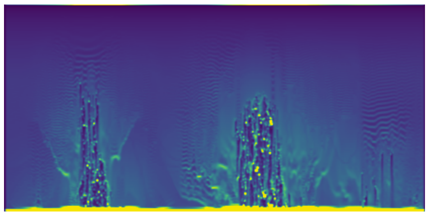




(a) 3 passes/frame, $\tau = 10^{-1}$, $t \approx 2.2$ (b) 10 passes/frame, $\tau = 10^{-2}$, $t \approx 1.8$ (c) 100 passes/frame, $\tau = 10^{-3}$, $t \approx 2.5$

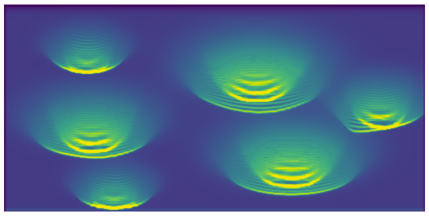
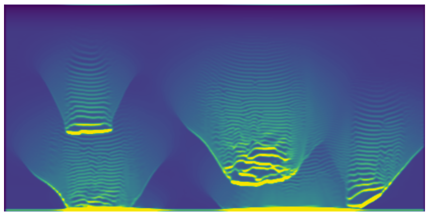
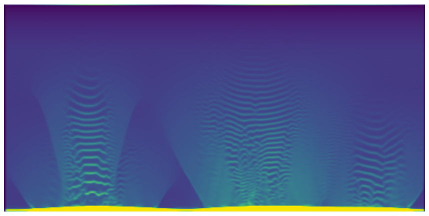




(d) 3 passes/frame, $\tau = 10^{-1}$, $t \approx 3$ (e) 10 passes/frame, $\tau = 10^{-2}$, $t \approx 2.6$ (f) 100 passes/frame, $\tau = 10^{-3}$, $t \approx 3.7$

Fig. 5. Flow for different time steps and number of passes/frame. The three columns illustrate the behaviour of the scheme (with comparable visual flow rate) for different hardware capabilities; low-end mobile device (left), high-end mobile/typical laptop (middle), and high-end desktop (right). Dewetting and droplet break-up is visible with the low-end settings, but the scheme remains stable.

(a) $\eta = 0$, $t \approx 1.2$ (b) $\eta = 0$, $t \approx 2.6$ (c) $\eta = 0$, $t \approx 4.3$

(d) $\eta = 20$, $t \approx 0.3$ (e) $\eta = 20$, $t \approx 1.2$ (f) $\eta = 20$, $t \approx 2.3$

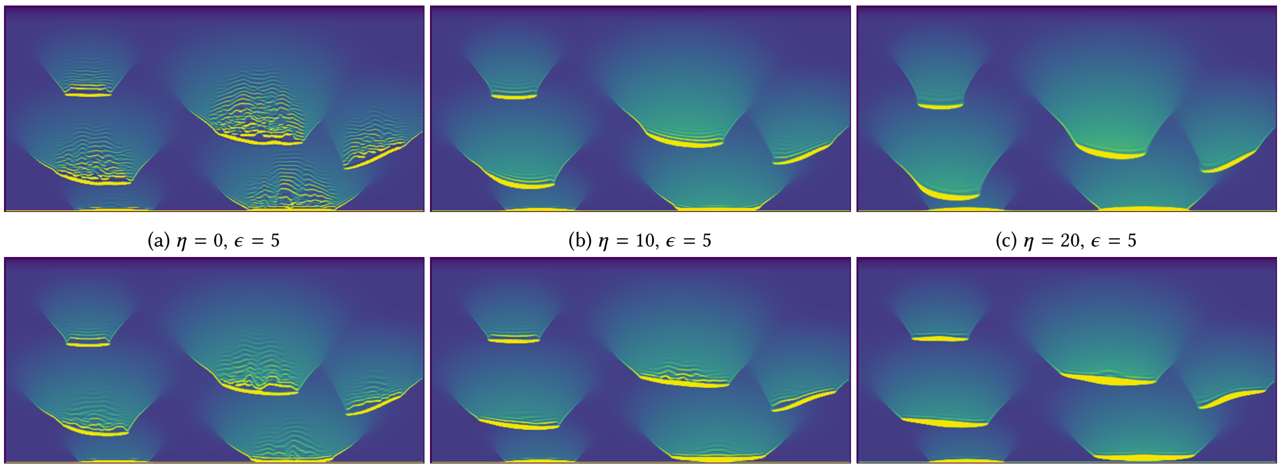
Fig. 6. Stabilising effect of the parameter η. Flow under strong gravity gradient $G = 100$. For very low η (top row), we observe dewetting and propagation of single-pixel droplets ("matrix effect"); due to its non-negativity and mass preservation properties the scheme remains stable regardless. Increasing η (bottom row) leads to a smoother flow even for a strong gravity gradient G.

A second limitation is that, although underlying surface features can be included by embedding their curvature into the external potential W (see Fig. 8b for an example), the scheme can not be applied as is on truly curved three-dimensional surfaces, as they can not be parametrised by a Cartesian grid. Furthermore, despite the three-dimensional appearance of the viscous fluid, especially when our scheme is coupled with a realistic rendering shader, it is fundamentally just a height-field attached to the surface. One could not use it to simulate fluid dripping off the surface for instance; some form of coupling with a particle system or a full-blown Navier-Stokes solver might be necessary for that.

4 RESULTS

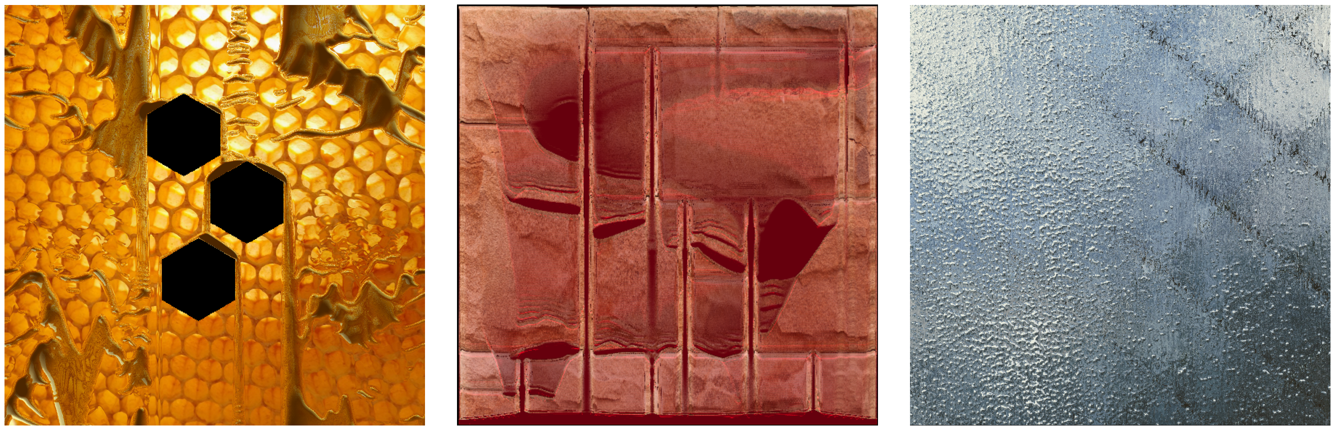
4.1 Simulations

The results in Fig. 9 - 12 present typical flow cases. The simulations were run in real time (at 10 iterations/frame, see sec. 4.2) on a 512x512 resolution, with the following parameters: $\tau = 2 \cdot 10^{-2}$, $\epsilon = 10$, $G = 10$ and $\eta = 2$, the gravity external potential $W(x, y) = Gy$ (for $(x, y) \in [0, 1]^2$) and the mobility $M(u_1, u_2) = \frac{2{u_1}^2{u_2}^2}{3(u_1+u_2)}$, and periodic boundary conditions. We rendered the fluid density with a colormap for clarity.


(a) $\eta = 0, \epsilon = 5$ (b) $\eta = 10, \epsilon = 5$ (c) $\eta = 20, \epsilon = 5$

(d) $\eta = 5, \epsilon = 5$ (e) $\eta = 5, \epsilon = 10$ (f) $\eta = 5, \epsilon = 20$

Fig. 7. Flow under different values of the parameters ϵ and η. Images captured at comparable times. Fixing η and varying ϵ (bottom row) leads to different typical size of local features (droplets, fronts), whereas fixing ϵ and varying η (top row) leads to an overall smoothening effect as η increases.


(a) Honey on a honeycomb. The fluid flows around the hexagonal dewetted areas.

(b) Wine flowing on bricks. The fluid concentrates in the spaces between the bricks.

(c) Rain on a window. Low viscosity and strong gravity liquids tend to form small droplets.

Fig. 8. Images rendered with a realistic refraction shader, showcasing various features of the interactive application (obstacles, interaction with the surface geometry, small scale features of the flow).

In Fig. 9, Gaussian concentrations of liquid of different sizes are placed in various positions and allowed to flow under the influence of gravity. There is residual fluid spread along the path of each Gaussian. This subsequently affects other Gaussians in its wake, as it is easier to flow along the path of higher concentration. In particular, the path of each Gaussian becomes biased in the direction of the preceding one. Note also the breaking up of the advancing concentrations by gravity into smaller waves - this effect becomes more prominent with stronger gravity and less so with higher values of ϵ and/or η (higher "viscosity"). This type of droplet interaction can lead to droplets merging, as can be seen in Fig. 10. The larger drop descends faster due to increased mobility of the fluid in the presence of more fluid. Eventually it catches up to the second drop and flows into it.

Another important feature of the model is that the fluid can not flow through dewetted areas, where the cells have zero density, which act as obstacles. As can be seen in Fig. 11, this leads to a concentration of the fluid at the top of the obstacles, until a way around them can be found. A sequence of obstacles, as in Fig. 12, can lead to a cascade of waterfall-like flows.

The images in Fig. 8 are representative of what a user might see while using the scheme in an interactive manner, with the output rendered with the refraction/caustics shader described in sec. 3.5. In the left-most image (Fig. 8a), the fluid has to flow around a set of hexagonal obstacles. Relatively high values of $\epsilon = 10$, compared to the gravity $G = 10$, assisted by high refractive indices in the refraction shader, give the appearance of a viscous fluid, such as honey. In the middle image (Fig. 8b), we have incorporated the effect that the shape of the underlying surface has on the fluid (see sec. 3.3). In this case, the fluid tends to accumulate and flow through the gaps between the bricks. Moreover, the fluid appears less viscous compared to Fig. 8a, since the ratio between ϵ and G is smaller ($\epsilon = 5$ and $G = 20$). In the final image (Fig. 8c), the viscosity is even weaker compared to the gravity ($\epsilon = 5$ and $G = 50$) which leads to the formation of very fine droplets.

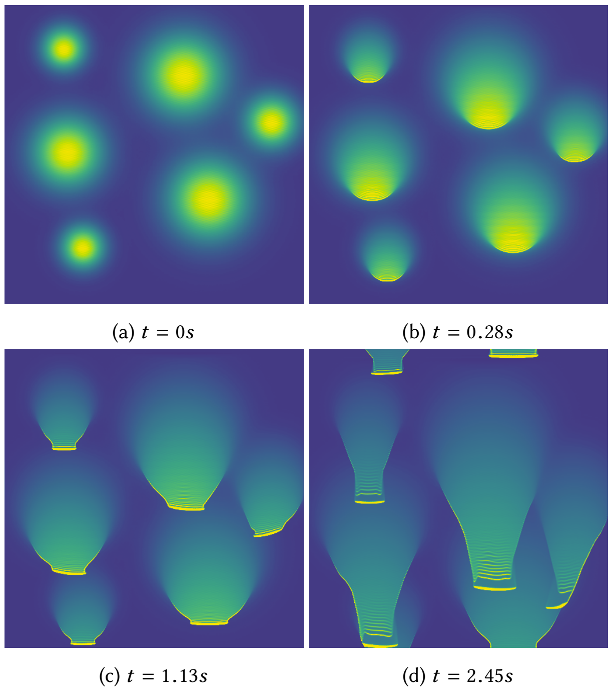

(a) $t = 0s$ (b) $t = 0.28s$ (c) $t = 1.13s$ (d) $t = 2.45s$

Fig. 9. A collection of Gaussians of various sizes. As they flow, they interact with each other's trail.

4.2 Performance

Our demo implementation using WebGL has demonstrated 60-120 fps performance with simulation resolutions of up to 512x512 (with Full HD rendering resolution) running around 10-20 iterations of the algorithm per time step on a PC with a dedicated Nvidia 1050 GTX graphics card. On various contemporary phones, none particularly high-end, we have seen 30 fps performance running around 10 iterations per time step with simulation and rendering resolutions of 256x256 pixels. It should be noted that one can improve performance, while maintaining reasonable visual effects, by simulating at a lower resolution than the rendering one. For reference, all the results shown in our paper and in the accompanying video do not exceed 512x512 simulation resolution.

4.3 User Interaction

Our demo application presents an interactive webpage, which implements our algorithm using WebGL. The webpage allows interactive

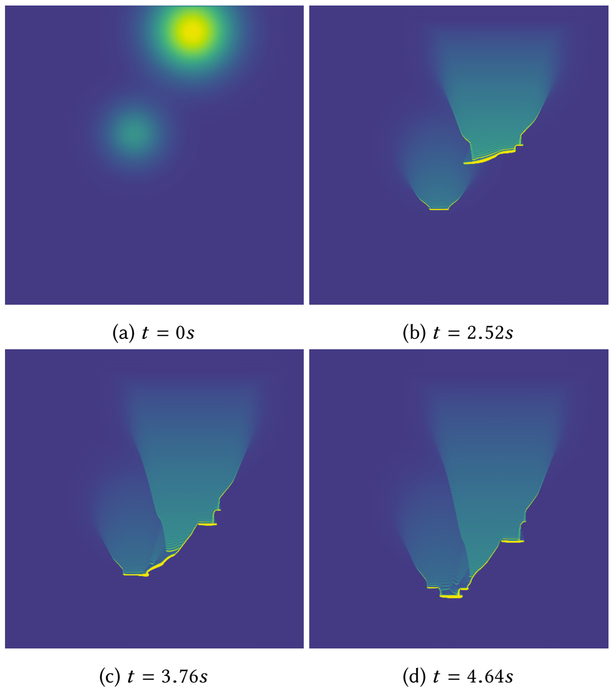

(a) $t = 0s$ (b) $t = 2.52s$ (c) $t = 3.76s$ (d) $t = 4.64s$

Fig. 10. Two droplets merging. As the larger droplet descends faster, it catches up to the lighter one and flows into it.

'spraying' of additional fluid using click/touch controls, and, on phones and tablets, control of the strength and direction of the gravity exerted on the fluid using the accelerometer. The application dynamically adjusts the number of iterations per frame to allow for a solid frame-rate on the device it is running on.

As mentioned in 3.6, fluid can not flow into regions where $u = 0$, meaning we can implement *obstacles* with no modifications to the algorithm. Our demo application allows the user to interactively *dewet* regions of the image, which the fluid then has to flow around (see Fig. 8a for example).

5 CONCLUSIONS AND FUTURE WORK

We described a scheme for the real-time simulation of viscous thin films on planar domains, that is efficient without compromising important theoretical properties. We also presented its implementation on parallel architectures, which allows for responsive user interaction together with realistic rendering, even on mobile devices.

Concerning potential future work, we would like to apply a similar scheme on non-Cartesian meshes. This would potentially allow us to simulate thin films on three-dimensional surface models with higher polygon count than possible with previous methods which require solving linear systems. Another possible extension is to include other physical effects in the simulation, such as evaporation or the inclusion of solubles, or other kinds of visual effects in the rendering, such as iridescence due to thin film interference. Finally, given the favorable practical and theoretical properties of the scheme, it would be interesting to see whether the same disciplined

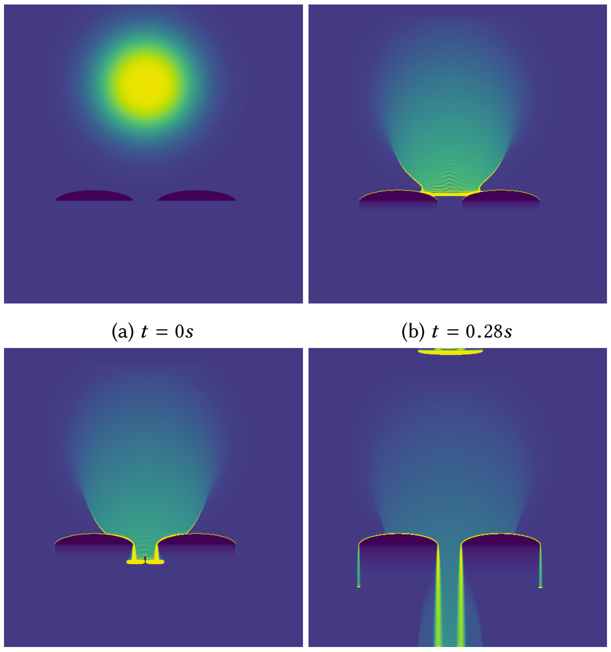

(a) $t = 0s$ (b) $t = 0.28s$ (c) $t = 1.13s$ (d) $t = 2.45s$

Fig. 11. A droplet is blocked by obstacles. The fluid accumulates, until it finds a way to flow around them.

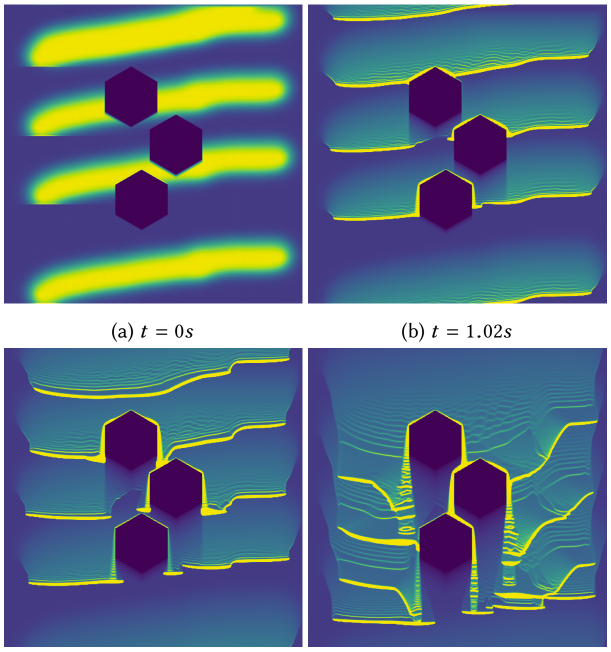

(a) $t = 0s$ (b) $t = 1.02s$ (c) $t = 2.33s$ (d) $t = 4.63s$

Fig. 12. Bands of fluid flow around and between obstacles. A cascade of waterfalls form, that are unstable and break up into drops.

approach could work on other problems of interest to the simulation community.

REFERENCES

Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. 2012. Discrete viscous sheets. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 113.

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete viscous threads. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 116.

A.L. Bertozzi, G. Grün, and T.P. Witelski. 2001. Dewetting films: bifurcations and concentrations. *Nonlinearity* 14, 6 (2001), 1569.

Robert Bridson. 2015. *Fluid simulation for computer graphics, 2ND EDITION.* A K Peters, Ltd.

Mark Carlson, Peter J Mucha, R Brooks Van Horn III, and Greg Turk. 2002. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation.* ACM, 167–174.

Zhili Chen, Byungmoon Kim, Daichi Ito, and Huamin Wang. 2015. Wetbrush: GPU-based 3D painting simulation at the bristle level. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 200.

RV Craster and OK Matar. 2009. Dynamics and stability of thin liquid films. *Reviews of modern physics* 81, 3 (2009), 1131.

E. De Giorgi and L. Ambrosio. 2006. New problems on minimizing movements. In *Ennio De Giorgi selected papers.* Springer, 699–714.

Prashant Goswami, Philipp Schlegel, Barbara Solenthaler, and Renato Pajarola. 2010. Interactive SPH simulation and rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 55–64.

Günther Grün and Martin Rumpf. 2000. Nonnegativity preserving convergent schemes for the thin film equation. *Numer. Math.* 87, 1 (2000), 113–152.

Mark J Harris. 2005. Fast fluid dynamics simulation on the GPU.. In *SIGGRAPH Courses.* 220.

Adrian RG Harwood and Alistair J Revell. 2018. Interactive flow simulation using Tegra-powered mobile devices. *Advances in Engineering Software* 115 (2018), 363–373.

D Jackson and J Gilbert. 2018. *WebGL 2.0 Specification.* Technical Report. Khronos Group. https://www.khronos.org/registry/webgl/specs/latest/2.0/.

Serafim Kalliadasis, Christian Ruyer-Quil, Benoit Scheid, and Manuel García Velarde. 2011. *Falling liquid films.* Vol. 176. Springer Science & Business Media.

Egor Larionov, Christopher Batty, and Robert Bridson. 2017. Variational stokes: a unified pressure-viscosity solver for accurate viscous liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 101.

Octavio Navarro-Hinojosa, Sergio Ruiz-Loza, and Moisés Alencastre-Miranda. 2018. Physically based visual simulation of the Lattice Boltzmann method on the GPU: a survey. *The Journal of Supercomputing* (2018), 1–27.

Alexander Oron, Stephen H Davis, and S George Bankoff. 1997. Long-scale evolution of thin liquid films. *Reviews of modern physics* 69, 3 (1997), 931.

F. Otto. 2001. The geometry of dissipative evolution equations: the porous medium equation. *Comm. in Partial Differential Equations* 26, 1-2 (2001), 101–174.

M. Rumpf and O. Vantzos. 2013. Numerical gradient flow discretization of viscous thin films on curved geometries. *Math. Models and Methods in Applied Sciences* 23, 05 (2013), 917–947.

Aviv Segall, Orestis Vantzos, and Mirela Ben-Chen. 2016. Hele-shaw flow simulation with interactive control using complex barycentric coordinates.. In *Symposium on Computer Animation.* 85–95.

Jacco H Snoeijer and Bruno Andreotti. 2013. Moving contact lines: scales, regimes, and dynamical transitions. *Annual review of fluid mechanics* 45 (2013).

Tuur Stuyck, Fang Da, Sunil Hadap, and Philip Dutré. 2017. Real-Time Oil Painting on Mobile Hardware. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 69–79.

Orestis Vantzos, Omri Azencot, Max Wardeztky, Martin Rumpf, and Mirela Ben-Chen. 2017. Functional Thin Films on Surfaces. *IEEE transactions on visualization and computer graphics* 23, 3 (2017), 1179–1192.

Huamin Wang, Gavin Miller, and Greg Turk. 2007. Solving general shallow wave equations on surfaces. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Eurographics Association, 229–238.

C. Yuksel and J. Keyser. 2009. Fast real-time caustics from height fields. *The Visual Computer* 25, 5-7 (2009), 559–564.

Liya Zhornitskaya and Andrea L Bertozzi. 1999. Positivity-preserving numerical schemes for lubrication-type equations. *SIAM J. Numer. Anal.* 37, 2 (1999), 523–555.

Bo Zhu, Minjae Lee, Ed Quigley, and Ronald Fedkiw. 2015. Codimensional non-Newtonian fluids. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 115.

Bo Zhu, Ed Quigley, Matthew Cong, Justin Solomon, and Ronald Fedkiw. 2014. Codimensional surface tension flow on simplicial complexes. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 111.