הטכניון
מכון טכנולוגי לישראל
TECHNION
Israel Institute of Technology

ספריות הטכניון
The Technion Libraries

בית הספר ללימודי מוסמכים ע"ש ארווין וג'ואן ג'ייקובס
Irwin and Joan Jacobs Graduate School

Real-time Simulation of Viscous Thin-Films

Saar Raz

Real-time Simulation of Viscous Thin-Films

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Saar Raz

Submitted to the Senate
of the Technion — Israel Institute of Technology
Sivan 5780 Haifa June 2020

This research was carried out under the supervision of Assoc. Prof. Mirela Ben-Chen, in the Center for Graphics and Geometric Computing, in the Faculty of Computer Science.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author's doctoral research period, the most up-to-date versions of which being:

Orestis Vantzos, Saar Raz, and Mirela Ben-Chen. Real-time viscous thin films. *ACM Trans. Graph.*, 37(6), December 2018.

Acknowledgements

I would like to thank Miri Ben-Chen and Orestis Vantzos for their continued support and collaboration throught the process.

I would like to deeply thank the Technion for the funding of this research, the publication and the accompanying trips.

Contents

List of Figures

Abstract

The intricate behavior of viscous thin films has fascinated physicists, mathematicians and engineers for many years; stable discrete schemes for simulation of the effect have been proposed, yet with the advent of mobile devices and graphics hardware the question of the feasibility of a real-time simulation of the effect arises. In our research we explore novel formulations and discrete schemes and attempt to simulate this effect at real-time rates while preserving physical fidelity; we propose a novel discrete scheme for simulating the effect on planar 2D surfaces: 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 2D 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. We also explore and show initial results using a discrete scheme derived from a novel formulation of the effect on curved surfaces, which allows simulating thin-films on curved 3D surfaces at real-time speeds with preserved visual fidelity.

Chapter 1

Introduction

The intricate behavior of viscous thin films has fascinated physicists, mathematicians and engineers for many years [ODB97, CM09]. 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 [VAW+17]. Schemes that can run at interactive rates [GSSP10], 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* [ODB97], 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 [Ott01], 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

Figure 1.1: Images captured during real-time simulation of viscous fluids.

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 [Bri15] 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.

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 [ODB97], and Craster et al. provide a more recent review [CM09]. The behavior of the contact line between the fluid and the dewetted region has also generated much research interest [SA13], as has the behavior of a thin layer on an inclined plane [KRQSV11].

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. [ZB99, GR00]). 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. [Ott01]). 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. [CMVHIT02], to the most recent treatment by Larionov et al. [LBB17], 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 [BAV$^+$10] and sheets [BUAG12] and Lagrangian co-dimensional methods that represent the fluid using a simplicial complex [ZQC$^+$14, ZLQF15] 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 [WMT07] 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 [RV13] and velocity-based [VAW$^+$17] 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 [SVBC16]. 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 [Har05, GSSP10, NHRLAM18]. Furthermore, such interactive fluid simulations can now run on mobile devices [HR18], enabling the user to control the simulation in real-time, for instance via the touch interface, [CKIW15, SDHD17]. 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).

We also provide a simplified numerical scheme for viscous thin film simulation on curved domains, which shows significant performance benefits over previous methods while preserving visual fidelity.

Chapter 2

2D Method

2.1 Physics

Consider the classic thin film equation [ODB97]

$$\frac{\partial u}{\partial t} = \operatorname{div}(M(u)\nabla w), \quad M(u) = \frac{u^3}{3}, \quad w = W - \epsilon\Delta u \tag{2.1}$$

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 (2.1) in terms of the *flux* f:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \operatorname{div} f = 0, \\ &f = -M(u)\nabla(W - \epsilon\Delta u + \eta u), \\ &u \geq 0 \end{aligned} \tag{2.2}$$

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 (2.1) has the following properties: a) it preserves the *total mass* $\int u\,\mathrm{d}x$ and b) the *Dirichlet energy* $\frac{1}{2}\int|\nabla u|^2\,\mathrm{d}x$ (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 [BGW01] and numerical [GR00] 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 (2.1), is to take advantage of their *gradient flow* structure [Ott01], 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 [GA06], where a (constrained) minimization problem of the (abstract) form

$$u^{n+1} = \underset{u\in\mathcal{R}(u^n)\subset X}{\operatorname{argmin}} \left\{ \frac{1}{2\tau}\operatorname{dist}_X^2(u,u^n) + \mathcal{E}(u) \right\} \tag{2.3}$$

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:

$$(u^{n+1},\phi^{n+1}) = \underset{\substack{u\in X,\phi\in Y\\ u-u^n+\tau\,\mathcal{D}_{u^n}\phi=0}}{\operatorname{argmin}} \left\{ \frac{\tau}{2} g_{u^n}(\phi,\phi) + \mathcal{E}(u) \right\} \tag{2.4}$$

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 (2.2); one can choose the *flux* f as the auxiliary variable, so that $u - u^n + \tau\operatorname{div} f = 0$ and $g_{u^n}(f,f) = \int M(u^n)^{-1}|f|^2\,\mathrm{d}x$, 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 [RV13] and the velocity-based in [VAW+17] for instance. Our scheme is based on the flux formulation of the gradient flow for a suitable energy $\mathcal{E}(u)$:

$$
\begin{aligned}
(u^{n+1}, f^{n+1}) &= \operatorname*{argmin}_{(u,f)\in\mathcal{R}(u^n)} \left\{ \frac{\tau}{2} \int M(u^n)^{-1} |f|^2 \,\mathrm{d}x + \mathcal{E}(u) \right\} \\
\mathcal{E}(u) &= \int \frac{\epsilon}{2} |\nabla u|^2 + W(x)u + \frac{\eta}{2}|u|^2 \,\mathrm{d}x \\
\mathcal{R}(u^n) &= \{(u,f) \,|\, u - u^n + \tau \operatorname{div} f = 0,\, u \geq 0\}
\end{aligned}
\tag{2.5}
$$

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

$$
\begin{aligned}
u^{(0)} &= u^n \\
u^{(k)} &= u^{(k-1)} - \tau\lambda(u^{(k-1)}, f_k) \operatorname{div} f_k \\
u^{n+1} &= u^{(K)}
\end{aligned}
\tag{2.6}
$$

The gradient flow scheme (2.5) 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 (2.2), 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:

$$
\mathcal{E}_h(u_h) = \frac{\epsilon}{2h^2} \sum_{p\to q} |u_p - u_q|^2 + \sum_p W_p u_p + \frac{\eta}{2} \sum_p |u_p|^2 \tag{2.7}
$$

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 (2.5), we minimize the sum of a suitable norm of the flux f and the discrete energy (2.7). As per the fractional step scheme (2.6), 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:

$$\begin{aligned}
\min_{f\in\mathbb{R}} \Big\{ & \frac{\tau|f|^2}{2\,M(u_p,u_q)} + \frac{\epsilon}{2h^2} \sum_{p'\to q'} |\tilde{u}_{p'} - \tilde{u}_{q'}|^2 \\
& + (W_p\tilde{u}_p + W_q\tilde{u}_q) + \frac{\eta}{2}(|\tilde{u}_p|^2 + |\tilde{u}_q|^2) \Big\} \\
\tilde{u}_p &= u_p - \frac{\tau}{h} f, \quad \tilde{u}_p \geq 0 \\
\tilde{u}_q &= u_q + \frac{\tau}{h} f, \quad \tilde{u}_q \geq 0
\end{aligned}$$

The sum in the middle term is over all the edges between p and q and their neighbours (and each other) (see Fig. 2.1).

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

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

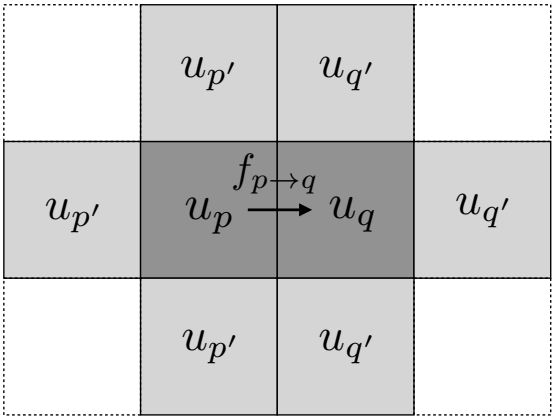


Figure 2.1: Flux $f_{p\to q}$ between two adjacent cells p and q. The flux depends on the values in the (5-cell) neighborhood of the two cells. See also Fig. 2.3a.

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.

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

$$\begin{aligned} f &= -\frac{M(u_p, u_q)}{\theta\, h}\Big\{W_q - W_p - \epsilon\left((\Delta_h u)_q - (\Delta_h u)_p\right) + \eta\left(u_q - u_p\right)\Big\} \\ f_{p\to q} &= \max(-\tfrac{h}{\tau}u_q, \min(f, \tfrac{h}{\tau}u_p)) \end{aligned} \tag{2.8}$$

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

$$\begin{aligned} u_p &\leftarrow u_p - \tfrac{\tau}{h} f_{p\to q} \\ u_q &\leftarrow u_q - \tfrac{\tau}{h} f_{q\to p} = u_q + \tfrac{\tau}{h} f_{p\to q} \end{aligned} \tag{2.9}$$

This is a rather straightforward discretization of (2.2), 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.

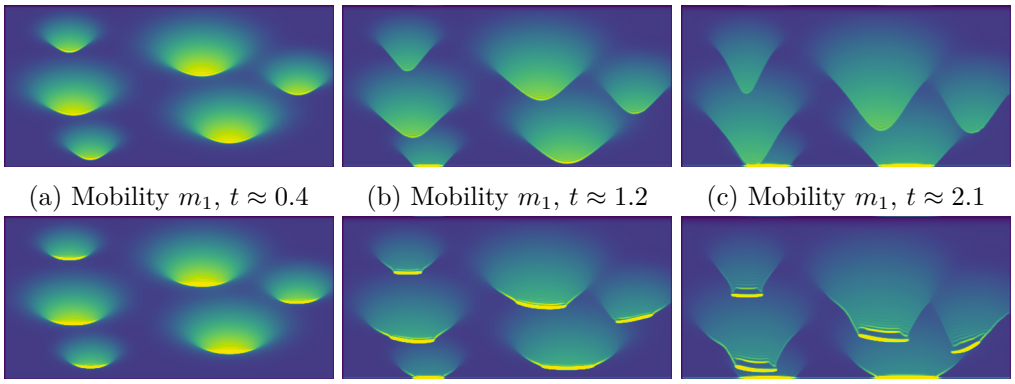
The *discrete mobility* $M(\cdot,\cdot)$ can be defined in many ways, as long as it is symmetric, $M(u_1, u_2) > 0$ for any positive $u_1 \neq u_2$, and $M(u, u) = \frac{u^3}{3}$. There are good theoretical arguments [GR00] in favour of the discrete mobility $M(u_1, u_2) = \frac{2u_1^2 u_2^2}{3(u_1+u_2)}$, 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. 2.2).

The local update (2.8)-(2.9) 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 (2.8). 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



(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, $t \approx 2.1$

Figure 2.2: 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.

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:

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

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. 2.3a. The horizontal edges are divided into 4 sets, and likewise for the vertical, for a total of 8 passes. Fig. 2.3b 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 $(1, 0)$ 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. 2.3a. 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

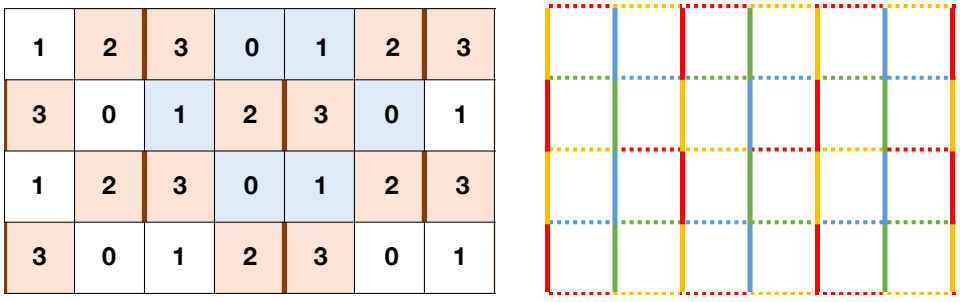
opposite direction $(-d_i, -d_j)$ (lines 4-9 in alg. 1). This ensures that both cells in a pair calculate the same flux in magnitude, but with opposite signs (since ij and $i'j'$ are effectively exchanged in line 14 of the alg.). Finally, only the cell pairs with parity 2 or 3 are allowed to update their values (see again Fig. 2.3a). 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*.

2.5 Implementation

2.6 WebGL Implementation

We based our implementation on WebGL [JG18], 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.



(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.

Figure 2.3: Partitioning of edges into passes

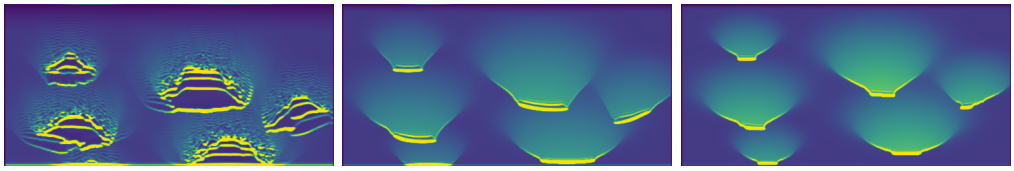
2.7 Dynamic Time Stepping

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.

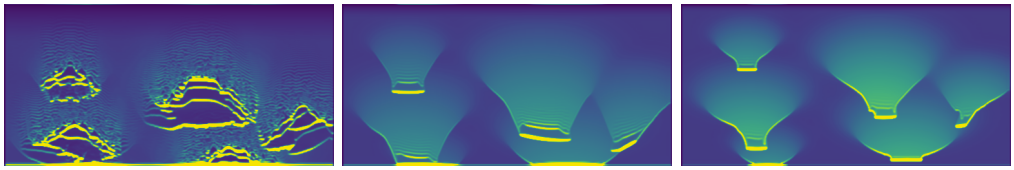
The basic idea behind the dynamic time stepping is that each full (visiting edges of all parities and directions) relaxation pass of the algorithm does a certain amount of work towards propagating the fluid proportional to the time step, i.e. $\tau \sim$ *work/pass*. A uniform movement rate then corresponds to a fixed total amount of work per second:

$$\tau \cdot \#_{\text{passes/frame}} \cdot \#_{\text{frames/sec}} \approx \text{work/sec} = const \tag{2.10}$$

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. 2.4 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. 2.5).

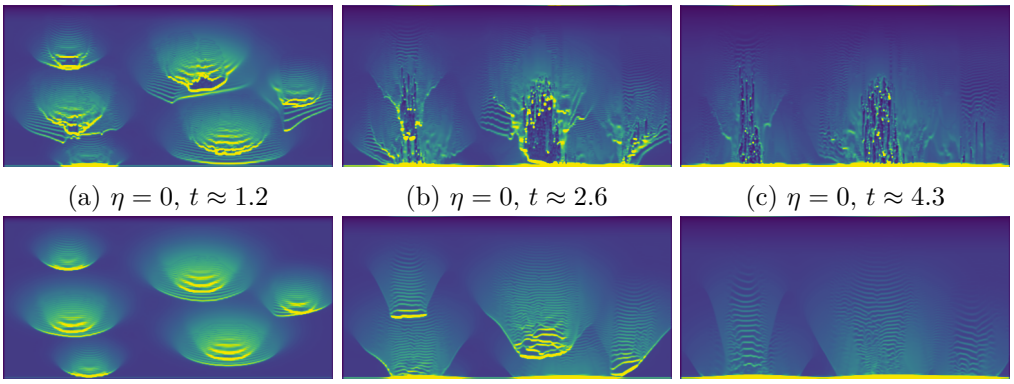


(a) 3 passes/frame, $\tau = 10^{-1}$, $t \approx 2.2$ (b) 10 passes/frame, $\tau = 10^{-2}$, $t \approx 1.8$ (c) 100 passes/frame, $\tau = 10^{-3}$, $t \approx 2.5$



(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, $\tau = 10^{-3}$, $t \approx 3.7$

Figure 2.4: 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) $\eta = 20$, $t \approx 2.3$

Figure 2.5: 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.

2.8 Driving the Flow via the External Potential

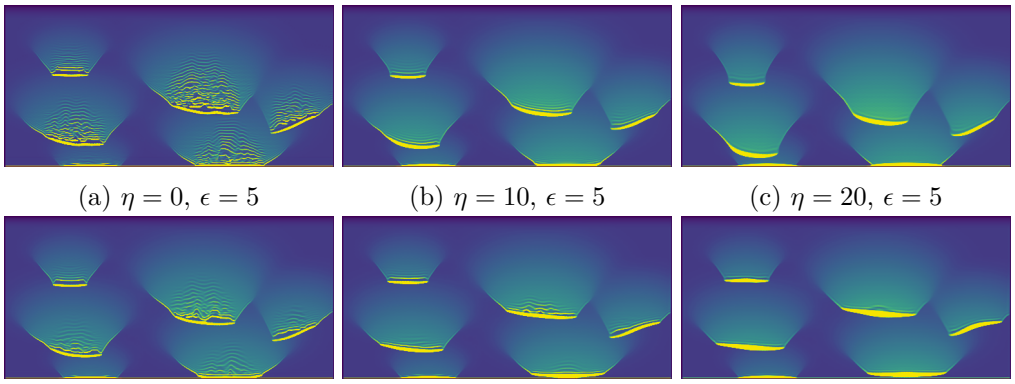
As the fluid tends to flow against the gradient of the external potential, i.e. from areas of high W to areas of low W, the external potential can be used to drive the fluid around the domain. The basic setting is a simple linear gradient, which corresponds to constant gravity. Its effect can be seen 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 [VAW+17], 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. 2.7b for an example of this.

2.9 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. 2.5) by penalising large concentrations of fluid. Intuitively, both parameters make the fluid appear more viscous ("thick"), although they are not equivalent; Fig. 2.6 illustrates the effect of these parameters.



(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) $\eta = 5$, $\epsilon = 20$

Figure 2.6: 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.

2.10 Rendering

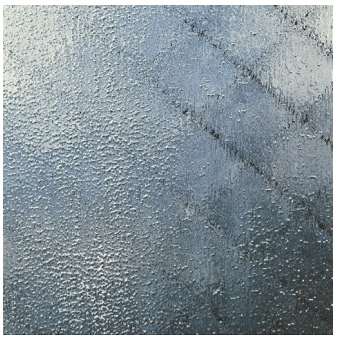
Our algorithm outputs a *height map* representing the fluid mass at each pixel. In our demonstration application we implemented a basic refraction shader with normals calculated from the height map, along with illumination based on *caustics* using the algorithm presented by [YK09].

2.11 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. 2.16).

(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.

Figure 2.7: 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).

2.12 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.

A second limitation is that, although underlying surface features can be included by embedding their curvature into the external potential W (see Fig. 2.7b 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.

2.13 Results

2.14 Simulations

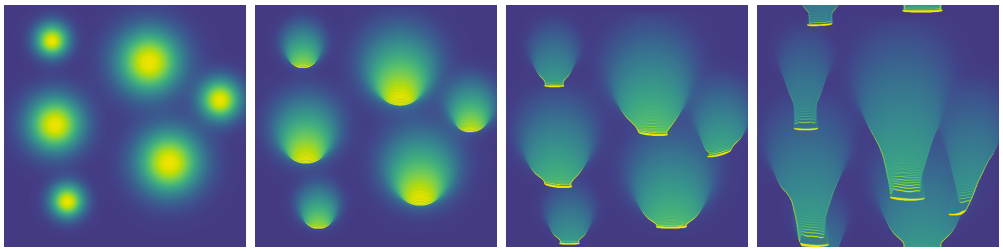
The results in Fig. 2.8 - 2.11 present typical flow cases. The simulations were run in real time (at 10 iterations/frame, see sec. 2.15) on a 512x512 resolution, with the following

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.

In Fig. 2.8, 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. 2.9. 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.

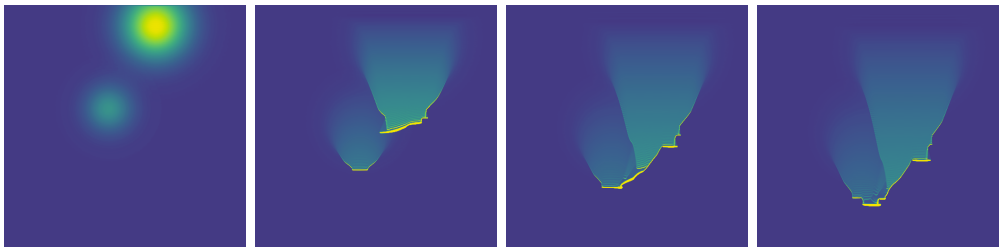
Another important feature of the model is that the fluid can not flow through dewetted areas, where the cells have zero density, which act as obstacles. As can be seen in Fig. 2.10, this leads to a concentration of the fluid at the top of the obstacles, until a way around them can be found. A sequence of obstacles, as in Fig. 2.11, can lead to a cascade of waterfall-like flows.

The images in Fig. 2.7 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. 2.10. In the left-most image (Fig. 2.7a), 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. 2.7b), we have incorporated the effect that the shape of the underlying surface has on the fluid (see sec. 2.8). 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. 2.7a, since the ratio between ϵ and G is smaller ($\epsilon = 5$ and $G = 20$). In the final image (Fig. 2.7c), the viscosity is even weaker compared to the gravity ($\epsilon = 5$ and $G = 50$) which leads to the formation of very fine droplets.



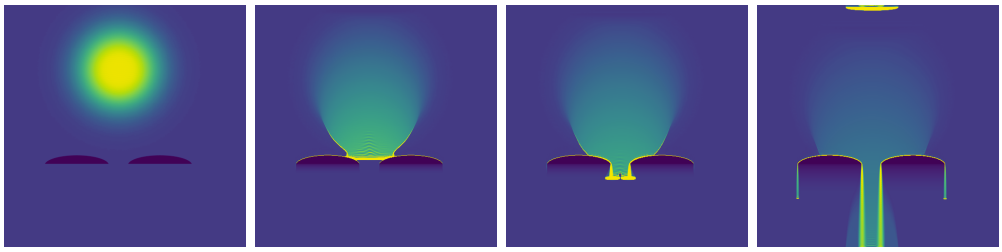
(a) $t = 0s$ (b) $t = 0.28s$ (c) $t = 1.13s$ (d) $t = 2.45s$

Figure 2.8: A collection of Gaussians of various sizes. As they flow, they interact with each other's trail.



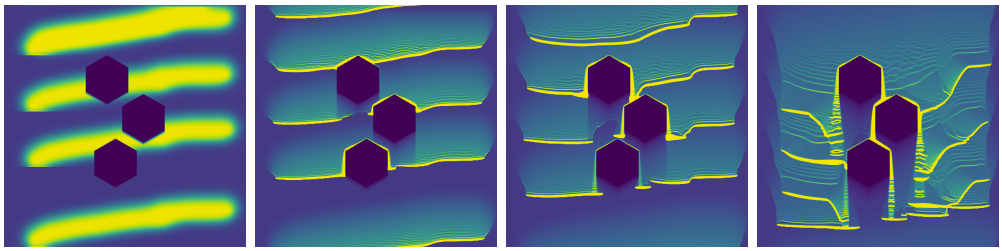
(a) $t = 0s$ (b) $t = 2.52s$ (c) $t = 3.76s$ (d) $t = 4.64s$

Figure 2.9: Two droplets merging. As the larger droplet descends faster, it catches up to the lighter one and flows into it.



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

Figure 2.10: 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) $t = 4.63s$

Figure 2.11: Bands of fluid flow around and between obstacles. A cascade of waterfalls form, that are unstable and break up into drops.

2.15 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.

2.16 User Interaction

Our demo application presents an interactive webpage, which implements our algorithm using WebGL. The webpage allows interactive '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 2.11, 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. 2.7a for example).

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

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

Chapter 3

3D Method

We continued to investigate achieving real-time simulations of thin-films on non-planar 3D surfaces.

3.1 Application of the 2D method

The method presented in previous sections operates on regular rectangular grids of cells; this yields a natural partitioning of the cells into independent passes, and furthermore provides a natural mapping to GPU hardware, which naturally handles rectangular arrays. The same method is not easily applicable to the more general case of general meshes, as stated in 2.12 - as those may not be parameterizable by a Cartesian grid.

3.2 Simplified thin-film formulation approach

Instead of adjusting the 2D method for 3D, we instead set out to reformulate a simplified version of the general flux-based thin-film equation on surfaces [RV13], attempting to simplify the equations in a manner that preserves visual fidelity, while gaining performance benefits.

3.2.1 Thin-film formulation for curved domains

The derivation for the lubrication model of thin films on curved domains as derived in [RV13] gives the following equations

$$\begin{aligned} &\frac{du}{dt} = \operatorname{div}(M(u)\nabla w) \\ &w = -H - gz - \varepsilon T u - \varepsilon \Delta u \\ &M(u) = \frac{1}{3}u^3 id + \frac{\varepsilon}{6}u^4(Hid - S) \end{aligned} \tag{3.1}$$

(3.1) contains a few additional terms compared to the classical thin-films equations (2.1). Namely, the definition of w now incorporates the surface curvature (mean curvature H, gaussian curvature K, $T = H^2 - 2K$ and the shape operator S): the external potential w incorporates the mean curvature H as well as the external gravity (represented by gz where g governs the strength of gravity and z is the 'up' facing coordinate of the vertices). The term εTu is a destabilizing term which serves as a higher order correction. The mobility $M(u)$ is also expanded to account for curvature with the second term.

3.2.2 Simplified formulation

The full form of this model has been given a full, stable discretization and optimization scheme in [VAW+17], but the resulting simulation did not manage to operate at real-time performance. In our attempt to create a real-time simulation of the effect, we would derive a discrete scheme for a simplified version of the above equations, by first dropping several high-order correction terms. Namely, we drop the destabilizing $-\varepsilon Tu$ term and the higher order corrections applied to the mobility. The resulting equations obtained are as follows:

$$\begin{aligned} \frac{du}{dt} &= \operatorname{div}(M(u)\nabla w) \\ w &= -H - gz - \varepsilon\Delta u \\ M(u) &= \frac{1}{3}u^3 \end{aligned} \tag{3.2}$$

The resulting equation is identical to the classical thin-film equation (2.1) with the external potential W incorporating the surface curvature and gravity.

3.2.3 Discretization

In order to discretize the simplified formulation over a discrete triangle mesh $M = \langle \mathcal{V}, \mathcal{F}, \mathcal{E} \rangle$', we would need a discretization of the mobility, and a suitable set of discrete operators div, ∇ and Δ. We use the set of discrete operators derived in [VAW+17], defined as follows:

$$\operatorname{div} = -G_{\mathcal{V}}^{-1} \operatorname{grad}^T G_{\mathcal{F}} \Delta = -\operatorname{div} \nabla. \tag{3.3}$$

$G_{\mathcal{V}} \in \operatorname{diag}(\mathbb{R}^{|\mathcal{V}|})$ contains the areas of the vertices along the diagonal, and $G_{\mathcal{F}} \in \operatorname{diag}(\mathbb{R}^{|\mathcal{F}|})$ contains the areas of the faces along the diagonal. $grad \in \mathbb{R}^{3|\mathcal{F}| \times |\mathcal{V}|}$ is the gradient operator, where the resulting gradients are represented as follows: the first $|\mathcal{F}|$ values along the diagonal are the X coordinates of the gradient vectors, the next $|\mathcal{F}|$ values are the Y coordinates of the gradient vectors, followed by the $|\mathcal{F}|$ Z coordinate values. Please see [BKP+10] for the explicit formulae for $G_{\mathcal{V}}$, $G_{\mathcal{F}}$, grad. The resulting discrete equation for a general triangular mesh with $|\mathcal{V}|$ vertices and $|\mathcal{F}|$ triangle faces

is as follows:

$$\frac{u - u^k}{\tau} = \operatorname{div} M(u^k)\nabla(-H - gz - \varepsilon\Delta u) \tag{3.4}$$

Where $u, u^k \in \mathbb{R}^{|\mathcal{V}|}$ are the fluid density at each vertex at the current and previous timestep respectively, τ represents the timestep size, $\operatorname{div} \in \mathbb{R}^{|\mathcal{V}|\times 3|\mathcal{F}|}$, $\nabla \in \mathbb{R}^{3|\mathcal{F}|\times|\mathcal{V}|}$, $\Delta \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, $H \in \operatorname{diag}(\mathbb{R}^{|\mathcal{V}|})$ is the mean curvature at each vertex, $z \in \mathbb{R}^{|\mathcal{V}|\times 1}$ contains the z coordinate of each vertex, and $M(u^k) \in \operatorname{diag}(\mathbb{R}^{3|\mathcal{F}|})$ is a diagonal matrix representing the mobility across each face of the mesh. We opted for a piecewise-constant mobility function across each face, equal to the average mobility of the 3 vertices

$$\forall f \in \mathcal{F} : M(u_k)_{f,f} = M(u_k)_{2f,2f} = M(u_k)_{3f,3f} = \frac{1}{3}\sum_{v\in f}\frac{1}{3}(u_v^k)^3$$

The mobility values are listed 3 times across the diagonal to match our discrete operators. Rearranging the terms we get the $|\mathcal{V}| \times |\mathcal{V}|$ linear system:

$$(I + \tau\varepsilon \operatorname{div} M(u^k)\nabla\Delta)u = u^k + \tau \operatorname{div} M(u^k)\nabla(-H - gz) \tag{3.5}$$

3.3 Implementation

We implemented a solver for (3.5) in C++, using the Eigen library ([GJ+14]) for linear algebra operations, libigl ([JP+18]) for triangle mesh visualization and I/O, and [EKS07] for calculating the mean curvature. The solution is presented in an interactive environment and allows for dynamic introduction of fluid by clicking on the mesh. We used the BiCGStab solver ([VdV92]) implemented in Eigen, which proved effective in achieving relatively low error in a small number of iterations. Fig. 3.2 shows example outputs from the implementation.

3.4 Stability adjustments

Our initial implementation of the scheme was not stable, and needed a few post and pre-processing adjustments in order to preserve stability.

3.4.1 Clamp at 0

The implicit scheme we derived does not enjoy the strong non-negativity guarantees of our 2D method, so we have to implicitly cap the values at 0 to avoid negative values from emerging. This needed adjustment also eliminates our guarantee of mass-preservation, for capping negative values at 0 would increase the total mass.

3.4.2 Mobility capping

Running the simulation for a while would eventually have the fluid concentrate at points of low pressure, namely points of highly negative curvature (creases and holes), and places of low gravity potential, i.e. the 'bottom' of the mesh. At those points, the film's thickness would start to accumulate. The rising fluid level is theoretically problematic as the lubrication approximation no longer holds, and so the simulation's perceived effect will diverge from physics. Numerically, the mobility around points of high fluid density will begin accumulating, quickly 'sucking' its surroundings and causing large negative values to accumulate. This will lead to the mass climbing due to the clamp at 0, and eventually destabilize the simulation with the mass rising exponentially, eventually causing NaNs to appear, breaking down the simulation. This phenomenon also causes the simulation to appear to slow down when larger values start to accumulate. This is most likely caused because the higher mobility values begin to dominate the equation, effectively causing cells with lower mobility to progress slower as the higher mobility ones have a larger effect on the energy. In order to fix this, we artificially clamp the maximum mobility at any given point at some fixed value (1 in our experiments). This has no physical justification but will also not take effect until large concentrations of fluid are formed, at which point the lubrication approximation does not hold in any case.

3.5 Initial performance optimizations

3.5.1 Initial solver guess

Starting with the fluid density at the previous timestep as an initial guess for the solver proved effective in lowering the error achieved by iterations.

3.5.2 Constant number of solver iterations

Solving with a constant, small number of iterations allows us to achieve a stable framerate and a built-in way to balance fidelity vs performance. A comparison of results given different iteration limits can be seen in Fig. 3.3.

3.5.3 Matrix-free system

Since our bi-conjugate solver's only use of the system matrix is multiplying by a column vector, we can multiply the vector by the matrix components separately without having to store the entire system matrix in memory. This gives a 10X performance improvement.

3.5.4 Vertex locality

Sorting the mesh's vertices in a way that increases locality would increase the memory locality of our matrix operations and might positively affect performance. After attempting to sort the vertices by locality over our set of test meshes, we've discovered an 8% improvement between randomly sorted vertices and vertices sorted by locality. The results shown henceforth do not rely upon vertex locality reorderings, as the meshes used have adequate vertex locality as provided, and further improving it proved ineffective performance-wise.

3.5.5 Results

In Fig. 3.3 we show the average frames per second and average iterative solver error per frame for 1, 3, 10 and unbounded solver iterations for a variety of meshes with this method. We use the relative residual error $|Ax - b|/|b|$ to convey how far we are from the exact solution of the equation on average, giving a sense of the accuracy being sacrificed per frame. It can be noted that running the simulation with a low solver iteration maximum resulted in a 'slower' progression of the fluid; Similar to the problem we observed in the 2D method, when running a smaller number of iterations, each frame contributes to the reduction of energy but does not truly converge therefore 'weakening' the timestep τ. To compensate for this, we adjust simulations with lower iteration counts with an increased timestep, to achieve a uniform apparent rate of progression, and allowing us to better qualitatively compare the different settings. With the adjusted time-steps, it can be seen that the higher iteration count simulations show increased formation of droplets and waveforms compared to the low iteration simulations.

3.6 Performance bottleneck

With the above optimizations in place, we identified that the performance bottleneck still resides in the multiplication of the system matrix by the column vector during solving. The system matrix originates from the fourth-order diffusion term that governs surface tension in the fluid. Being a fourth-order term, it is relatively dense and therefore more expensive to handle. Its reliance upon the fluid density of the previous iteration prevents us from pre-calculating the inverse, which would allow us to greatly speed up the process.

3.6.1 Second-order diffusion

One way to improve performance would be to replace the fourth order diffusion term with a second order diffusion term - the latter should be more sparse and therefore be cheaper to multiply. The resulting system would be:

$$(I + \tau\varepsilon \operatorname{div}\operatorname{div} M(u^k)\nabla)u = u^k + \tau \operatorname{div} M(u^k)\nabla(-H - gz) \tag{3.6}$$

3.6.2 No-mobility diffusion

The fourth order diffusion term is the only term in the system matrix which depends upon the previous timestep, forcing us to recalculate the system matrix at each timestep. Were that term not to depend on the mobility, we would be able to invert the system matrix in preprocessing, and gain a vast speedup. We chose to simply drop the mobility term from the fourth-order diffusion term, effectively running the simulation as if all mobility values are 1, resulting in the following formulation:

$$(I + \tau\varepsilon \operatorname{div} \nabla\Delta)u = u^k + \tau \operatorname{div} M(u^k)\nabla(-H - gz) \tag{3.7}$$

3.6.3 Results

Fig. 3.1 shows the original (fourth order with mobility) diffusion method derived from the original formulations alongside the two proposed simplifications, all run to a similar error threshold. The no-mobility diffusion method demonstrates a more blurry result, with some noise in the form of blurred dry areas; the result, however, is roughly similar in structure, and is achieved at a much faster frame-rate. The second-order diffusion alternative converges the quickest (most likely due to the lower order of the resulting equation), but displays results rather different from the other two methods, with larger droplets unable to form; it showcases a quicker frame-rate than the fourth-order method as expected, yet the performance gain is less significant than the one gained from the no-mobility diffusion method. Fig 3.6 shows how the 'finger' effect is weakened when using either of the simplified diffusion alternatives. Table. 3.6.3 compares performance and solver error (relative residual error, as before) of the fourth-order method with and without mobility, and also to the results presented in [VAW+17], where available. It can be seen that the no-mobility method achieves similar error results at a much higher framerate and at fewer iterations compared to the regular method at 10 iterations. In fig. 3.7 we compare the output of our real-time method with a result presented in [VAW+17], to better illustrate the qualitative difference between the methods' outputs.

Mesh	# Vertices	3 iterations		10 iterations		6 iterations (no-mobility)		[VAW+17]
		FPS	Error	FPS	Error	FPS	Error	FPS
rounded_cube_10k	9.1K	70.1	3.E-02	29.7	3.E-03	209.6	7.E-03	
sphere_s3	10.2K	57.7	6.E-03	25.3	4.E-04	179.7	6.E-04	
torus_s3	10K	46.7	1.E-02	26.2	8.E-04	187.2	3.E-03	
ellipsoid_s3	10.2K	56.4	4.E-02	29.3	3.E-02	178.9	1.E-02	
moomoo_s3	16.7K	29.5	4.E-06	14.1	4.E-11	103.0	1.E-14	12.5
bunny	16.6K	23.0	8.E-02	11.3	2.E-02	90.3	1.E-02	
vase	21.8K	22.5	2.E-04	11.6	2.E-05	71.2	5.E-06	
bumpy_sphere_s3	21.1K	18.9	9.E-06	9.6	7.E-10	63.3	2.E-09	
pensatore_s4	27.7K	13.5	1.E-01	6.6	3.E-02	39.2	4.E-02	1.21
moai_s3	21.8K	19.7	9.E-02	9.4	3.E-02	33.7	4.E-02	
bunny_s4	38.3K	9.0	9.E-03	4.4	3.E-04	27.3	4.E-04	2.06
torus_s4	40K	12.7	4.E-02	6.4	1.E-02	37.6	1.E-02	0.926
sphere_s4	41K	10.9	4.E-02	5.5	4.E-02	36.1	3.E-02	0.607
bumpy_plane_s4	40.4K	12.1	2.E-01	6.5	1.E-01	37.6	1.E-01	1.464
scherk_surface_s3	40.4K	15.6	6.E-02	6.4	4.E-02	35.1	5.E-02	1.59
moomoo_s4d	37K	9.4	3.E-06	4.6	3.E-10	34.0	1.E-11	
pensatore_s5	50K	7.8	1.E-01	3.7	4.E-02	26.1	6.E-02	
fat_torus_s4	39.2K	10.3	3.E-02	5.1	2.E-02	33.7	1.E-02	
bumpy_sphere_s5	49.1K	6.2	6.E-05	3.2	1.E-07	26.1	1.E-06	
moai_s7	89.1K	2.8	1.E-01	1.5	5.E-02	13.1	1.E-01	0.321
sphere_s5	163.8K	2.1	4.E-02	0.8	3.E-01	8.3	2.E-01	

Table 3.1: Measurements of FPS and error with 3 different configurations. Measurements were made on an Intel Core-i7 7700HQ CPU. See Fig. 3.4 for scatter plot.

Fourth order diffusion: 10 iterations, 3.38 frames/s, error=0.0003

Frame 1 Frame 50 Frame 100

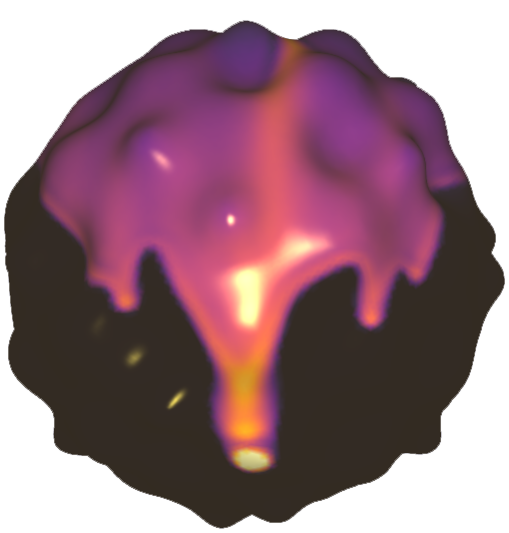
No-mobility diffusion: 6 iterations, 25.96 frames/s, error=0.0002

Frame 1 Frame 50 Frame 100

Second-order diffusion: 4 iterations, 11.49 frames/s, error=0.003

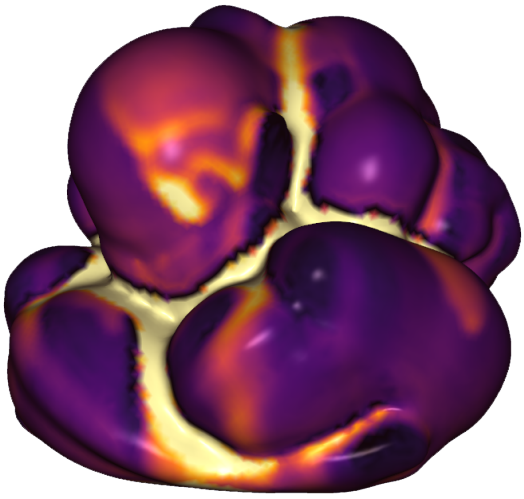
Frame 1 Frame 50 Frame 100

Figure 3.1: Comparison of simulation fidelity and performance with different diffusion methods, run to a similar error threshold per frame ($\varepsilon = 1., g = 20$, 38306 vertices). All error values are the relative residual error $|Ax - b|/|b|$ for the corresponding linear system.

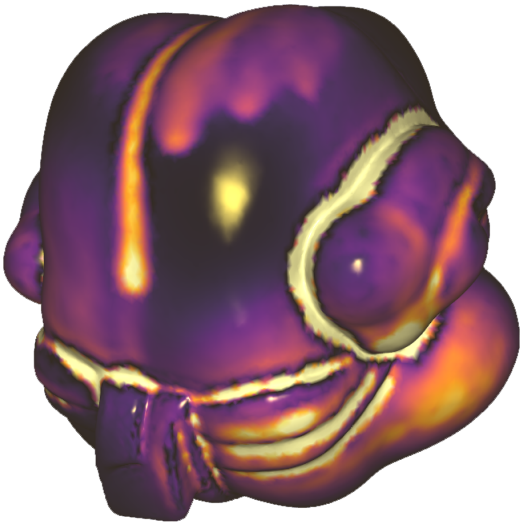


Bumpy Sphere (33.6K vertices), 4 iterations, $g = 17.24$, $\varepsilon = 2.6$, $\tau = 10^{-2}$, 11FPS

Moomoo (37K vertices), 3 iterations, $g = 20$, $\varepsilon = 3.4$, $\tau = 10^{-2}$, 11FPS



Pensatore (27K vertices), 3 iterations, $g = 310$, $\varepsilon = 5.4$, $\tau = 10^{-2}$, 14FPS



Pensatore (50K vertices), 3 iterations, $g = 120$, $\varepsilon = 4.9$, $\tau = 10^{-2}$, 9FPS

Figure 3.2: Examples of viscous thin-film flow on curved surfaces as generated in real-time by our implementation

1 iteration per frame - frames/s: 23.1, error: 0.299 ($\tau = 0.3$)

Frame 1 Frame 50 Frame 100

3 iterations per frame - frames/s: 12.4, error: 0.183 ($\tau = 0.1$)

Frame 1 Frame 50 Frame 100

10 iterations per frame - frames/s: 4.76, error: 0.003 ($\tau = 0.0333..$)

Frame 1 Frame 50 Frame 100

Convergence (avg. 112 iterations)- frames/s: 0.5, error$\propto 10^{-16}$ ($\tau = 0.03$)

Frame 1 Frame 50 Frame 100

Figure 3.3: Comparison of simulation fidelity and performance with different BiCGStab iteration count limits per frame ($\varepsilon = 1., g = 10$, 38306 vertices). All error values are the relative residual error $|Ax - b|/|b|$ for the linear system in equation 3.5.

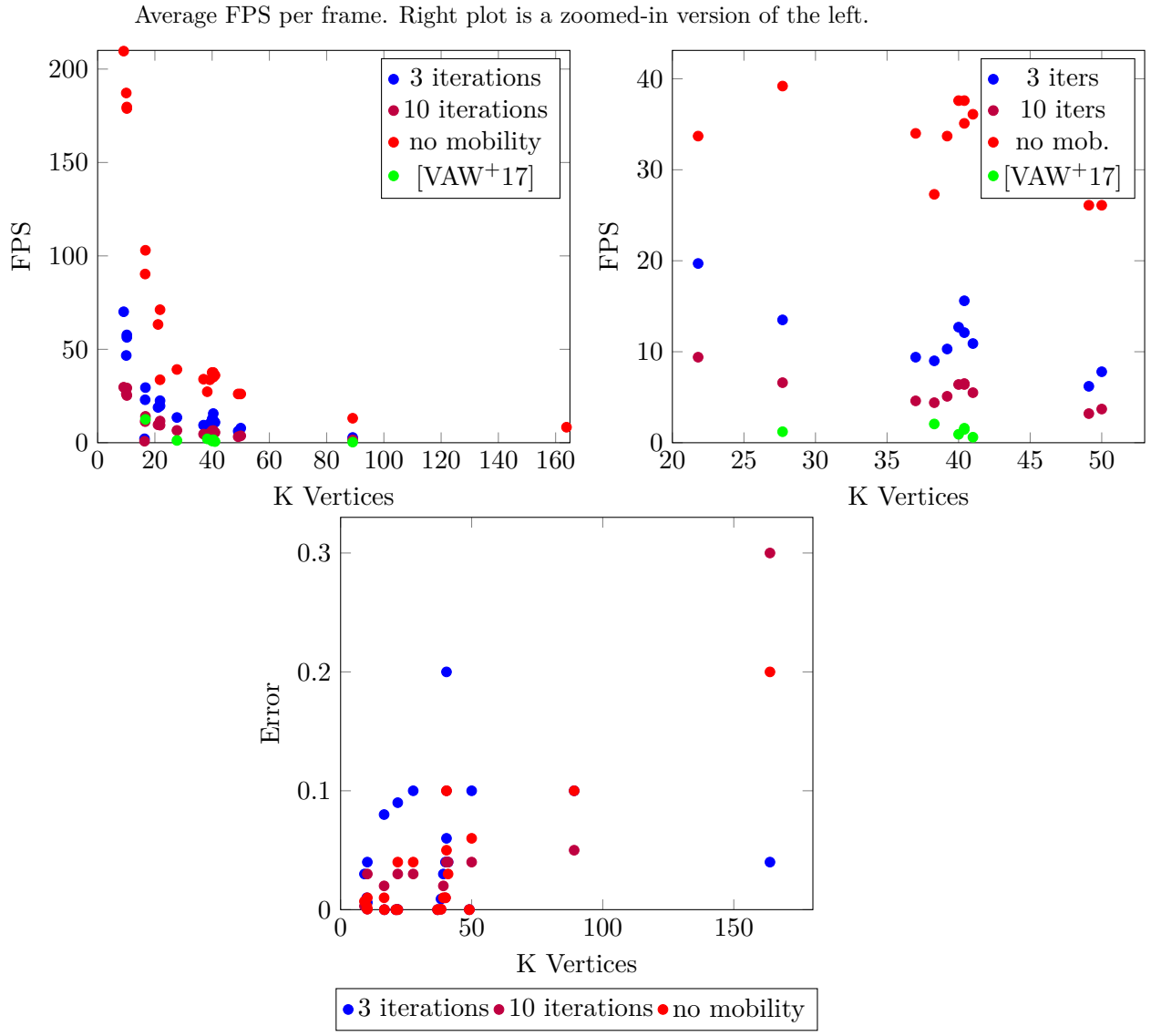


Figure 3.4: Plots of performance data as shown in Table 3.6.3

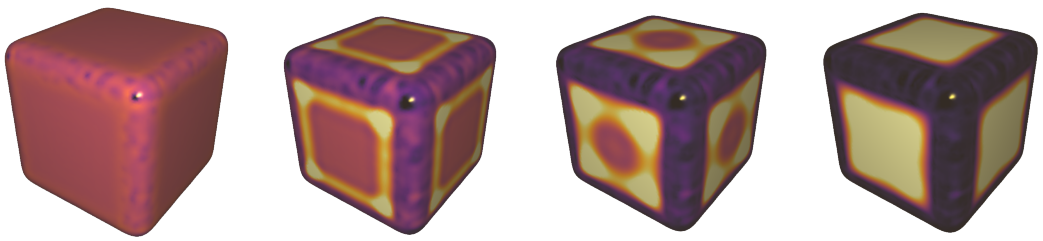


Figure 3.5: In the absence of gravity, surface tension balances the fluid on top of every face, compare with [VAW+17] Fig. 10

4th order diffusion 2nd order diffusion Mobility-free diffusion

Figure 3.6: With regular 4th order diffusion, the mobility function prevents most fluid from flowing into dewetted areas, favoring areas already filled with fluid; this allows for the creation of 'fingers' as shown. When using second order diffusion, only coarser fingers are able to form due to the lower order of the diffusion equation. When using mobiliy-free diffusion, the fingers are smeared out, as the diffusion of the fluid is not impeded across the dewetted area.

Figure 3.7: Visual comparison with Fig. 2 of [VAW+17]. The real-time result was captured at 11FPS on a 38K vertex bunny mesh.

Chapter 4

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. We presented directions for achieving a real-time simulation of viscous thin films on curved domains as well, with a scheme that allows natural control of the fidelity-performance tradeoff, and can achieve visually pleasing results at real-time framerates.

Concerning potential future work, we would like to further enhance performance by using a multigrid approach. 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 approach could work on other problems of interest to the simulation community.

Bibliography

[BAV+10] Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. Discrete viscous threads. *ACM Transactions on Graphics (TOG)*, 29(4):116, 2010.

[BGW01] A.L. Bertozzi, G. Grün, and T.P. Witelski. Dewetting films: bifurcations and concentrations. *Nonlinearity*, 14(6):1569, 2001.

[BKP+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[Bri15] Robert Bridson. *Fluid simulation for computer graphics, 2ND EDITION*. A K Peters, Ltd., 2015.

[BUAG12] Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. Discrete viscous sheets. *ACM Transactions on Graphics (TOG)*, 31(4):113, 2012.

[CKIW15] Zhili Chen, Byungmoon Kim, Daichi Ito, and Huamin Wang. Wetbrush: Gpu-based 3d painting simulation at the bristle level. *ACM Transactions on Graphics (TOG)*, 34(6):200, 2015.

[CM09] RV Craster and OK Matar. Dynamics and stability of thin liquid films. *Reviews of modern physics*, 81(3):1131, 2009.

[CMVHIT02] Mark Carlson, Peter J Mucha, R Brooks Van Horn III, and Greg Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 167–174. ACM, 2002.

[EKS07] Derek Nowrouzezahrai Evangelos Kalogerakis, Patricio Simari and Karan Singh. Robust statistical estimation of curvature on discretized surfaces. *Proceedings of the Eurographics/ACM Siggraph Symposium on Geometry Processing (SGP '07), pp. 13-22*, 2007.

[GA06] E. De Giorgi and L. Ambrosio. New problems on minimizing movements. In *Ennio De Giorgi selected papers*, pages 699–714. Springer, 2006.

[GJ+14] Gael Guennebaud, Benoit Jacob, et al. Eigen: a c++ linear algebra library. *URL http://eigen. tuxfamily. org, Accessed*, 22, 2014.

[GR00] Günther Grün and Martin Rumpf. Nonnegativity preserving convergent schemes for the thin film equation. *Numerische Mathematik*, 87(1):113–152, 2000.

[GSSP10] Prashant Goswami, Philipp Schlegel, Barbara Solenthaler, and Renato Pajarola. Interactive sph simulation and rendering on the gpu. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–64. Eurographics Association, 2010.

[Har05] Mark J Harris. Fast fluid dynamics simulation on the gpu. In *SIGGRAPH Courses*, page 220, 2005.

[HR18] Adrian RG Harwood and Alistair J Revell. Interactive flow simulation using tegra-powered mobile devices. *Advances in Engineering Software*, 115:363–373, 2018.

[JG18] D Jackson and J Gilbert. Webgl 2.0 specification. Technical report, Khronos Group, 2018. https://www.khronos.org/registry/webgl/specs/latest/2.0/.

[JP+18] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/.

[KRQSV11] Serafim Kalliadasis, Christian Ruyer-Quil, Benoit Scheid, and Manuel García Velarde. *Falling liquid films*, volume 176. Springer Science & Business Media, 2011.

[LBB17] Egor Larionov, Christopher Batty, and Robert Bridson. Variational stokes: a unified pressure-viscosity solver for accurate viscous liquids. *ACM Transactions on Graphics (TOG)*, 36(4):101, 2017.

[NHRLAM18] Octavio Navarro-Hinojosa, Sergio Ruiz-Loza, and Moisés Alencastre-Miranda. Physically based visual simulation of the lattice boltzmann method on the gpu: a survey. *The Journal of Supercomputing*, pages 1–27, 2018.

[ODB97] Alexander Oron, Stephen H Davis, and S George Bankoff. Long-scale evolution of thin liquid films. *Reviews of modern physics*, 69(3):931, 1997.

[Ott01] F. Otto. The geometry of dissipative evolution equations: the porous medium equation. *Comm. in Partial Differential Equations*, 26(1-2):101–174, 2001.

[RV13] M. Rumpf and O. Vantzos. Numerical gradient flow discretization of viscous thin films on curved geometries. *Math. Models and Methods in Applied Sciences*, 23(05):917–947, 2013.

[SA13] Jacco H Snoeijer and Bruno Andreotti. Moving contact lines: scales, regimes, and dynamical transitions. *Annual review of fluid mechanics*, 45, 2013.

[SDHD17] Tuur Stuyck, Fang Da, Sunil Hadap, and Philip Dutré. Real-time oil painting on mobile hardware. In *Computer Graphics Forum*, volume 36, pages 69–79. Wiley Online Library, 2017.

[SVBC16] Aviv Segall, Orestis Vantzos, and Mirela Ben-Chen. Hele-shaw flow simulation with interactive control using complex barycentric coordinates. In *Symposium on Computer Animation*, pages 85–95, 2016.

[VAW+17] Orestis Vantzos, Omri Azencot, Max Wardeztky, Martin Rumpf, and Mirela Ben-Chen. Functional thin films on surfaces. *IEEE transactions on visualization and computer graphics*, 23(3):1179–1192, 2017.

[VdV92] Henk A Van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.

[WMT07] Huamin Wang, Gavin Miller, and Greg Turk. Solving general shallow wave equations on surfaces. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 229–238. Eurographics Association, 2007.

[YK09] C. Yuksel and J. Keyser. Fast real-time caustics from height fields. *The Visual Computer*, 25(5-7):559–564, 2009.

[ZB99] Liya Zhornitskaya and Andrea L Bertozzi. Positivity-preserving numerical schemes for lubrication-type equations. *SIAM Journal on Numerical Analysis*, 37(2):523–555, 1999.

[ZLQF15] Bo Zhu, Minjae Lee, Ed Quigley, and Ronald Fedkiw. Codimensional non-newtonian fluids. *ACM Transactions on Graphics (TOG)*, 34(4):115, 2015.

[ZQC+14] Bo Zhu, Ed Quigley, Matthew Cong, Justin Solomon, and Ronald Fedkiw. Codimensional surface tension flow on simplicial complexes. *ACM Transactions on Graphics (TOG)*, 33(4):111, 2014.

משמעותי. הדיסקרטיזציה שיצאה הצריכה מאיתנו לפתור מערכת משוואות לינארית בכל צעד של ההדמיה, לשם כך השתמשנו בפותר איטרטיבי, שאיפשר הפעלה לכמות איטרציות מוגבלת, ובכך לשמור על זמן ריצה קבוע בכל צעד זמן, תוך שימור כמות שגיאה ואיכות ויזואלית לבחירתינו. הניסוח החדש בשילוב עם כמה שיפורי ביצועים נוספים אפשר לנו להגיע לביצועי זמן-אמת סבירים על מודלים בעלי כ40000 קודקודים, אך עדיין זיהינו צוואר-בקבוק משמעותי הטמון בחישוב המערכת הלינארית הנפתרת בכל צעד. אנו מציעים כמה שיטות אפשריות לפתרון צוואר הבקבוק בנסיון לשמור על מהימנות המשוואות המפושטות למשוואה המקורית.

ומאפשרים למשתמש לשלוט על הסימולציה בזמן אמת, למשל דרך ממשק מגע. אף על פי כן, למיטב ידיעתינו לא קיימת כיום הדמיית זמן-אמת של סרטים דקים של נוזל צמיגי, שכמוה אנו מדגימים.

אנו מציעים סכמה נומרית להדמיית משוואת הסרט הדק על משטחים שטוחים, בשילוב כוח כבידה וכוחות חיצוניים נוספים. הסכמה שלנו מבוססת על שינוי קל בקירוב השימון, בו הנוזל מיוצג כפונקציית גובה מעל התחום המישורי. השינוי שלנו מוסיף איבר ריבועי למשוואה, המצייב את הזרימה תוך שמירה על מהימנות ויזואלית של הסימולציה. הדיסקרטיזציה שלנו בזמן ובמרחב מבוססת על גישת זרימת הגרדיאנט, בה ההתקדמות של הנוזל על פני ציר הזמן נשלטת ע"י ירידה בכיוון הגרדיאנט של פונקציונאל אנרגיה, ומבטיחה שימור מדוייק של מסת הנוזל הכוללת, ואי-שליליות של פונקציית הגובה. לבסוף, הדיסקרטיזציה שלנו היא מקומית, ולכן מקלה על מקבול על הGPU, ללא צורך בגישות יקרות לזכרון. מימשנו את את השיטה שלנו בWebGL, והדגמנו את יכולתה לרוץ בקצב אינטראקטיבי, ולאפשר למשתמש להתערב בזרימת הנוזל ע"י הוספה של נוזל נוסף, מכשולים ושליטה בכיוון וחוזק כוח הכבידה.

הדיסקרטיזציה של הסכמה שלנו נעשית על גבי רשת ריבועית של תאים, בה הערך בכל תא הוא צפיפות הנוזל באותו התא. המבנה הרגולרי של הרשת, בשילוב עם הלוקאליות של מבנה השבלונה של הסכמה, מאפשרים לנו למפות את הסכמה באופן טבעי למעבד הגרפי, העובד באופן טבעי על מערכים מלבניים של ערכים. הדיסקרטיזציה של הסכמה שלנו ממדלת את אבולוציית הנוזל בעזרת פעולה בסיסית אחת המשמרת תכונות רצויות - זרימה של נוזל מתא אחד ברשת לתא השכן אליו. הפעולה הזו משמרת בהגדרה את סכום המסות בכלל התאים, מורידה את האנרגיה הכוללת ומאפשרת שליטה קלה על אי-השליליות של צפיפות הנוזל בכל תא (זאת בכדי למנוע כמות שלילית של נוזל בתאים מסויימים, תופעה שקורית לרוב בשיטות אימפליציטיות) ע"י הגבלת הנוזל המועבר בין תא לתא כך שלא יווצר נוזל שלילי. הפעולה האטומית הנ"ל נובעת מנגזרת חלקית של האנרגיה הכוללת ביחס לתא נוזל מסויים. לכן, העברה בודדת של נוזל מתא אחד לשכנו מהווה בעצם gradient descent ביחס לנגזרת חלקית מסויימת. שרשור הולך ונשנה של פעולות כאלו אחת אחרי השניה תהווה קירוב של ירידת גרדיאנט עבור כולן יחדיו. עובדה זו מאפשרת לנו לקדם את ההדמיה ע"י חלוקתה למעברים נפרדים, כשבכל מעבר נעביר נוזל בין תאים שאינם תלויים בתאים האחרים באותו מעבר. כך נחלק את כל מעברי הנוזלים ונבצע את ההעדכון של כל מעבר במקביל ע"י הפעלת shader במעבד הגרפי המעביר את הנוזלים בין כל זוג תאים ומרנדר את התוצאה לתמונה חדשה. בכל timestep נבצע את כל המעברים אחד אחרי השני כדי לאפשר לנוזל לעבור בכל הכיוונים. את השיטה על משטחים דו מימדיים מימשנו בליווי אפליקציה נוחה לאינטרקציה של משתמשים עם הנוזל. השיטה הוכיחה את עצמה אפקטיבית בהשגת קצבי זמן אמת במחשבים שולחניים עם מעבדים גרפיים מודרנים, כמו כן גם בטלפונים מודרניים (אם כי ברזולוציית הדמייה נמוכה יותר). במימוש על טלפונים המשתמש יכול לשלוט על כיוון ועוצמת הכבידה ע"י שינוי זווית הטלפון (ע"י שימוש באקסלרומטר). השיטה המוצגת עבור משטחים שטוחים דו מימדיים פועלת על רשת מלבנית ורגולרית, דבר המאפשר חלוקה קלה של התאים למעברים בלתי תלויים וכפועל יוצא מיפוי טבעי לחומרת מעבדים גרפיים. אותה השיטה לא מתמפה באופן ישיר למשטחים עקמומיים כלליים בתלת מימד, משום שאלה לרוב לא ממופים כרשת מלבנית. במקום להתאים את השיטה שלנו למשטחים שטוחים דו מימדיים לתלת מימד, בחרנו לנסות לנסח מחדש גרסא מפושטת של משוואת הסרט הדק על משטחים מבוססת השטף, במטרה לפשט את המשוואות בצורה המשמרת את המהימנות הויזואלית אך מאפשרת שיפורי ביצועים משמעותיים. לשם כך, ויתרנו על כמה איברים בעלי סדר גבוה במשוואה, שהשפעתן על ההדמיה לא ניכרת באופן

תקציר

במחקר אנו סוקרים את הסימולציה של סרטים דקים של נוזל צמיגי בזמן אמת, בתלת ובדו מימד; אנו מציעים סכמה דיסקרטית חדשה להדמיית האפקט על משטחים מישוריים דו-מימדיים: הסכמה מבוססת על ניסוח חדש של גישת ה gradient flow, שמובילה לדיסקרטיזציה מבוססת שגלוגות מקומיות הניתנות לחישוב קל ומהיר על המעבד הגרפי. השיטה שלנו משמרת מהימנות פיזיקלית, בכך שהיא משמרת את המסה הכללית, בשליטתה בפונקציית אנרגיה מתאימה ובשימור גובה נוזל אי-שלילי שהיא מבטיחה לאורך הסימולציה. בנוסף, היא מהירה מספיק בשביל לאפשר אינטראקציה בזמן אמת עם הזרימה, זאת בניגוד לשאר השיטות הקיימות לסימלוץ האפקט; כך אנחנו מאפשרים למשתמש לעצב בעצמו את המצב ההתחלתי ולשלוט בכוחות ובצפיפות הנוזל תוך כדי ריצת הסימולציה. בנוסף אנו סוקרים ומראים תוצאות ראשוניות מסכמה דיסקרטית שנגזרה מניסוח חדש של האפקט על משטחים תלת מימדיים עקמומיים, המאפשרת סימולציה בזמן אמת תוך שמירה על מהימנות ויזואלית.

המשוואות המייצגות את זרימתם של סרטים דקים של נוזלים נחקרו במשך שנים רבות, אמפירית כנומרית. מבחינה נומרית, סרטים דקים מדומים לרוב בעזרת קירוב השימון, פישוט של מודל נביה-סטוקס המבוסס על ההנחה שעובי הסרט הוא קטן מספיק. מדובר במשוואה דפרנציאלית חלקית מסדר רביעי, מה שמוביל לקשיים הנובעים מצורך בצעדי זמן קטנים מאוד לסכמות אקספליסיטיות. לחלופין, ניתן להתייחס לאבולוציה של הסרט כאל זרימת גרדיאנט, שיטה בה תנועת הנוזל מנוסחת כירידת גרדיאנט בהתאם לפונקציית אנרגיה כלשהיא בתוך מטריקה מסוימת, כאשר כאן המטריקה מקודדת את התנגדות הנוזל לתזוזה הנובעת מצמיגיותו. לסכמות כאלה דיסקרטיזצית זמן טבעית המשמרת את מבנה הזרימה, כגון המסה הכללית, והיא יציבה נומרית. הגישה שלנו מבוססת על שינוי קטן של קירוב השימון, שבעוד אין בו בסיס פיזיקלי, תורם ליציבות הזרימה תוך שמירה על אמינות ויזואלית של הפלט. בנוסף, אנו מספקים ניסוח מבוסס שטף שבניגוד לשיטות הקיימות מוביל לסכמה מקומית לחלוטין.

בקהילת הגרפיקה נוזלים צמיגיים הודמו בעבר במתכונת של זרימות שטח חופשיות, המדמות את המודל התלת מימדי המלא, ובכך לא מנצלות את המימדיות הנמוכה של סרטים מדקים. הדמיות מים רדודים בזמן אמת אחרות שקיימות כיום מאפשרות להדמות מבני גלים מורכבים, אך אינם מתאימים להדמיית נוזלים בעלי צמיגיות גבוהה, כגון דבש.

הדמיית נוזלים תמיד היתה משימה קשה חישובית, מה שהוביל לאתגרים שונים בשליטה בתנאי ההתחלה ובכוחות הפועלים בזמן ההדמיה, במיוחד בגרפיקה ממוחשבת, בה נרצה שהנוזל יוכוון בקלות ע"י אומן. בשנים האחרונות, מעבדים גרפיים נהיו כלי חשוב לחישוב יעיל יותר בישומים רבים, ביניהם הדמיית נוזלים. בנוסף, הדמיות אינטראקטיביות שכאלה רצות כיום גם על טלפונים ניידים,

המחקר בוצע בהנחייתו של פרופ/ח בן-חן מירלה, במרכז לגרפיקה וחישוב גאומטריה, בפקולטה למדעי המחשב.

חלק מן התוצאות בחיבור זה פורסמו כמאמרים מאת המחבר ושותפיו למחקר בכנסים ובכתבי-עת במהלך תקופת מחקר הדוקטורט של המחבר, אשר גרסאותיהם העדכניות ביותר הינן:

Orestis Vantzos, Saar Raz, and Mirela Ben-Chen. Real-time viscous thin films. *ACM Trans. Graph.*, 37(6), December 2018.

תודות

אני רוצה להודות למירי בן-חן ולאורסטיס ונצוס, על עזרתם והעבודה המשותפת לאורך הפרוייקט.

ברצוני להודות מקרב לב לטכניון על מימון המחקר, הפרסום והנסיעות הנלוות אליו.

סימולציה של סרטים דקים של נוזל צמיגי בזמן אמת

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במדעי המחשב

סער רז

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
סיוון התש"פ חיפה יוני 2020

סימולציה של סרטים דקים של נוזל צמיגי בזמן אמת

סער רז