SPH Fluids for Viscous Jet Buckling

Luiz Fernando de Souza Andrade [*], Marcos Sandim [*], Fabiano Petronetto [†], Paulo Pagliosa[‡] and Afonso Paiva[*]

[*]ICMC, USP, São Carlos
[†] Dept. of Mathematics, UFES, Vitória
[‡]FACOM, UFMS, Campo Grande

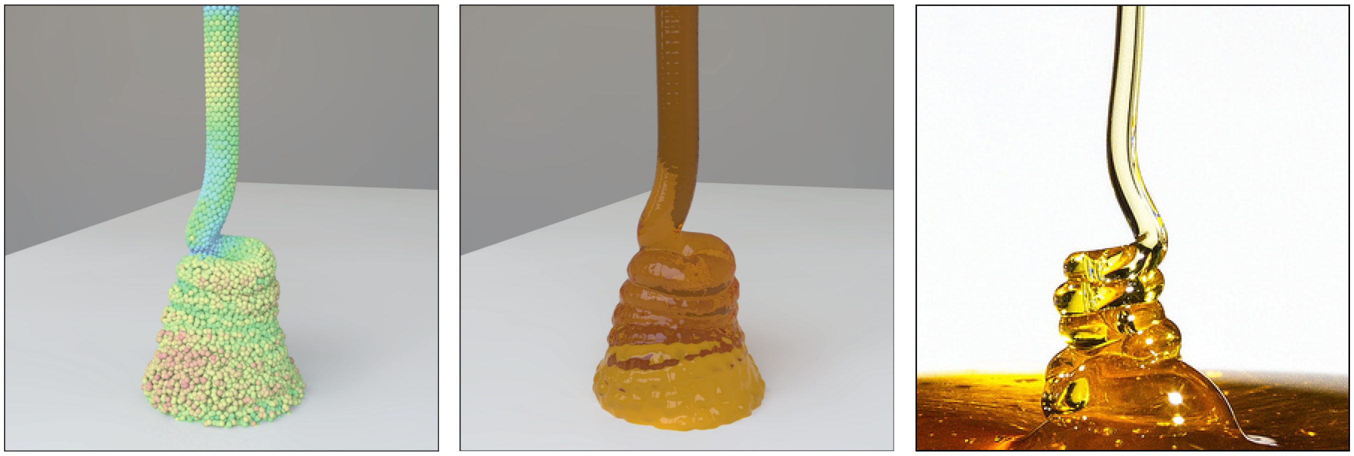


Fig. 1. The liquid rope coiling effect: our technique with SPH fluid particles (left), and the respective free surface (middle) and a real image of honey coiling (right) extracted from *Flickr* repository.

***Abstract*—We present a novel meshfree technique for animating free surface viscous liquids with jet buckling effects, such as coiling and folding. Our technique is based on Smoothed Particle Hydrodynamics (SPH) fluids and allows more realistic and complex viscous behaviors than the preceding SPH frameworks in computer animation literature. The viscous liquid is modeled by a non-Newtonian fluid flow and the variable viscosity under shear stress is achieved using a viscosity model known as Cross model. The proposed technique is efficient and stable, and our framework can animate scenarios with high resolution of SPH particles in which the simulation speed is significantly accelerated by using Computer Unified Device Architecture (CUDA) computing platform. This work also includes several examples that demonstrate the ability of our technique.**

***Keywords*-SPH fluids; jet buckling; viscous liquids; CUDA; computer animation.**

I. Introduction

A daily life example of viscous jet buckling is the coiling and folding of a thin thread of syrup or honey falling onto a spoon. The characteristic motion of a jet buckling is controlled by the balance among inertia, gravity and viscous forces that arise from the compressive stress caused by the impact of the fluid on a rigid surface. In the last years, Smoothed Particle Hydrodynamics (SPH) [1] has become a popular numerical meshfree tool for visually realistic animation of liquids [2]. However, simulating the complex free surface of a viscous jet buckling in an efficient and realistic way remains a big challenge for the previous SPH frameworks in computer animation. The difficulties are related to propose a variable viscosity model which has a non-linear dependence of the fluid's shear rate, an accurate and stable SPH approximation for viscous acceleration which involves second order derivatives of each component of the velocity field and enforcing boundary conditions suited to SPH.

In this paper, we present a novel meshfree technique based on SPH fluids for simulating viscous jet buckling behaviors. Our technique allows a wide range of realistic viscous effects of the free surface of liquids, such as coiling and folding, as shown in Figure 1. In order to capture the viscous behavior that is characteristic of jet buckling, the time interval between two consecutive frames needs to be very short, as discussed in Section III-C, thus increasing the number of time-steps along the simulation total time. Since the computation in each time-step is highly intense, but can be performed in parallel and independently for each SPH particle, the problem can be suitably mapped to graphics processing units (GPUs). We use the Computer Unified Device Architecture (CUDA) by NVIDIA due to its efficiency, object-oriented programming capability, easy integration with the development environment we have used, and availability of a lot of libraries and demos which accompany the CUDA toolkit. The adequacy of using CUDA for standard SPH fluids can be demonstrated by other implementations reported in the literature [3], [4]. In summary, the main contributions of this paper are:

Variable viscosity model. The viscous liquid is modeled as a non-Newtonian fluid flow and the variable viscosity is governed by a rheological model known as *Cross model* [5].

SPH viscous acceleration. We introduce a stable and robust SPH approximation of fluid's viscous acceleration using derivative operators of first order.

SPH jet buckling on CUDA. Using CUDA enables the animation of jet buckling scenes involving millions of SPH particles in affordable computational times, notably when compared to sequential processing, as showed by the experiments presented in Section IV, freeing the CPU for other tasks.

A. *Related work*

In order to better contextualize our approach and highlight its properties, we organize the existing frameworks for animating viscous jet buckling into two main groups, Eulerian mesh-based and Lagrangian meshfree-based methods.

Eulerian mesh-based. A seminal work in computer animation was introduced by Goktekin et al. [6]. They simulate solids and viscoelastic fluids with a small effect of buckling using an explicit grid-based method with viscosity transition between solid and non-Newtonian fluid controlled by a quasi-linear plasticity model. Batty and Bridson [7] developed an implicit and unconditionally stable method using marker-and-cell (MAC) grid. Although this method provides high-accuracy free surface boundary conditions, it is limited to viscous Newtonian fluids. Bergou et al. [8] proposed a discrete model for viscous threads using elastic rods to represent thin Newtonian liquid jets. Despite this method's realistic results, spurious results may occur when the jet becomes thick. Recently, this model was extend to discrete viscous thin sheets [9]. Batty and Houston [10] presented an adaptive tetrahedral mesh solver to animate Newtonian liquids with high viscosity. However, the level set surface generated by this method does not preserve temporal coherence due to the slow motion of the liquid. In computational physics literature, there are several papers using variations of generalized simplified MAC (GENSMAC) method to simulate viscous jet buckling in arbitrary 2D/3D domains with explicit [11], [12], [13] and implicit [14] free surface boundary conditions.

Lagrangian meshfree-based. SPH fluids have been applied with success in simulations of highly viscous liquids with variable viscosity [15], [16], [17], [18], [19]. However, none of these methods in computer animation have captured viscous buckling behavior. In computational physics, Rafiee et al. [20] used an incompressible version of SPH to simulate 2D jet buckling of non-Newtonian fluids, while Xu et al. [21], [22] extended the traditional weakly compressible SPH method to deal with 3D simulations. This paper is inspired in [21] and it improves that work in several ways: our stable SPH approximation of momentum equation does not require additional terms, artificial stress and artificial viscosity, to prevent particle clustering and unphysical behavior of free surface.

To our best knowledge, this paper is the first work to propose a meshfree framework to animate viscous jet buckling in computer animation literature.

II. Governing Equations

The governing equations for simulating fluid flow are derived from mass and momentum conservation laws. Lagrangian framework describes these laws from the viewpoint of an infinitesimally small fluid element, i.e., a particle. In this framework the mass conservation is naturally satisfied, since the particle mass is constant, then the total mass of the system is preserved. For weakly compressible fluids, the momentum equation can be written as follow:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \tau + \mathbf{g} \tag{1}$$

where t denotes the time, $\mathbf{v}$ the velocity field, ρ the density, p the pressure, $\mathbf{g}$ the gravity acceleration vector and τ the shear stress tensor.

Lagrangian formulation of Equation (1) represents the acceleration of a particle moving with the fluid flow. The term $-\frac{1}{\rho}\nabla p$ is related to particle acceleration due to pressure changes in the fluid. While, the term $\frac{1}{\rho}\nabla \cdot \tau$ describes the viscous acceleration due to friction forces caused by particles with different velocities. This last term plays a key role in viscous jet buckling animation.

Cross model

In order to animate a wide variety of buckling effects, Newtonian and non-Newtonian fluid flows are used in this paper. In particular, non-Newtonian fluids have non-linear dependence of the shear stress τ with respect to the rate-of-deformation tensor $\mathbf{D} = \nabla\mathbf{v} + (\nabla\mathbf{v})^{\top}$ as follows

$$\tau = \rho\, \nu(D)\, \mathbf{D}\,, \quad \text{with} \quad D = \sqrt{\tfrac{1}{2} \cdot \text{trace}\,(\mathbf{D})^2}\,. \tag{2}$$

The Cross model [5] is one of simplest and most used model for shear-thinning behavior, i.e., the fluid's viscosity decreases with increasing of the local shear rate D, thus the kinematic viscosity ν is defined as a function of D:

$$\nu\,(D) = \nu_\infty + \frac{\nu_0 - \nu_\infty}{1 + (KD)^n}\,, \tag{3}$$

where K and n are positive parameters, and ν_0, ν_∞ are the limiting values of the viscosity at low and high shear rates, respectively. The units of viscosity are m^2/s.

Assuming $K = 0$ in Equation (3), the non-Newtonian fluid model is simplified to a Newtonian fluid with constant kinematic viscosity ν_0.

III. Our Technique

There are several SPH frameworks to animate fluid flow. In our animation framework, we use a SPH version for weakly compressible fluids [23], extended by our method. A wide description of SPH fluids for graphics can be found in [2].

A. SPH fluids

The main idea of SPH in fluid flow simulation is to discretize the fluid by a set of particles where each particle i has attributes such as position $\mathbf{x}_i$, velocity $\mathbf{v}_i$, pressure p_i, mass m_i (constant across all particles) and density ρ_i. A scalar or vector attribute $f_i = f(\mathbf{x}_i)$ is updated using a SPH approximation

$$f_i = \sum_{j \in N_i} f_j \, W_h(x_{ij}) \frac{m_j}{\rho_j} \,, \tag{4}$$

where j indexes the particles lying in the neighborhood N_i of the particle i and $x_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$. The kernel function W_h is a radially symmetric, positive, smooth, compactly supported function and the value h defines the region of influence of W_h. Fluid attributes and the differential operators of the momentum equation (1) can be approximated in a similar manner as follows.

Density computation. We compute the density using the traditional direct summation derived from Equation (4) with *poly6* kernel presented in [23]:

$$\rho_i = \sum_{j \in N_i} m_j \, W_h(x_{ij}) \,. \tag{5}$$

This form preserves mass exactly without involving kernel derivatives.

Equation of state. In standard SPH frameworks, an incompressible fluid is approximated by a weakly compressible fluid. In other words, the pressure is computed directly from density through an equation of state. For simplicity, we choose an equation of state proposed by Morris et al. [24]:

$$p_i = c^2(\rho_i - \rho_0) \,, \tag{6}$$

where c is the speed of sound in the fluid and ρ_0 is a reference density. The values $c = \sqrt{1.5}\, m/s$ and $\rho_0 = 10^3\, kg/m^3$ are tuned out to be suitable for all experiments.

Pressure acceleration. We use Müller's SPH pressure gradient with *spiky* kernel [23]. The pressure acceleration for each particle is given by:

$$-\frac{1}{\rho_i}\nabla p_i = -\sum_{j \in N_i} m_j \frac{p_i + p_j}{2\rho_j} \nabla_i W_h(x_{ij}) \,. \tag{7}$$

This choice avoids numerical instabilities associated with particle clustering, because the SPH gradient does not vanish when x_{ij} becomes closer to zero.

Viscous acceleration. In order to compute the shear stress τ_i at particle i in Equation (2), we adopt the same SPH approximation as Paiva et al. [19] for the deformation tensor $\mathbf{D}_i = \nabla \mathbf{v}_i + \nabla \mathbf{v}_i^\top$ with:

$$\nabla \mathbf{v}_i = \sum_{j \in N_i} \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) \otimes \nabla_i W_h(x_{ij}) \,. \tag{8}$$

After updating shear stress at all particles, the viscous acceleration is approximated by:

$$\frac{1}{\rho_i} \nabla \cdot \tau_i = \sum_{j \in N_i} \left(\frac{\tau_i}{\rho_i^2} + \frac{\tau_j}{\rho_j^2} \right) \cdot \nabla_i W_h(x_{ij}) \,. \tag{9}$$

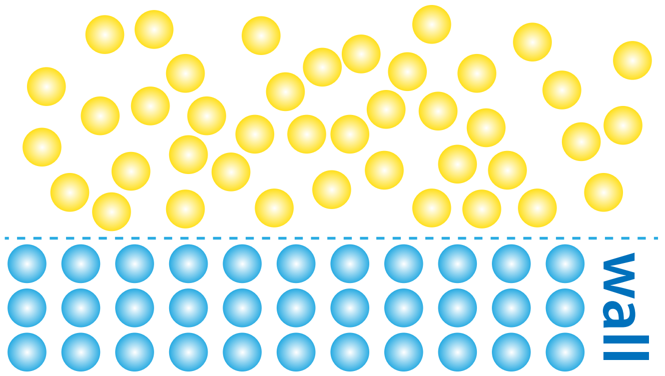


Fig. 2. SPH particles: fluid (yellow) and boundary (blue) particles.

B. Boundary conditions

We represent the solid wall boundaries by fixed virtual particles [25]. These boundary particles perform similar to the fluid particles and contribute to the SPH approximation of the fluid attributes. The density of the fluid particles is extrapolated for boundary particles using Equation (5). Then, the pressure is computed in the boundary particles directly from (6). The increasing pressure at boundary particles prevents the fluid particles from penetrating to the solid walls. In order to enforce the no-slip condition, the velocities of the boundary particles are set to zero throughout the entire simulation. Those particles are placed outside the computational domain with particle spacing equal to the initial particle spacing of the fluid particles. The wall boundaries are modeled by a few layers of the boundary particles as illustrated in Figure 2. The particle deficiency at wall boundaries is alleviated by taking the total width of boundary particle layers at least equal to the radius of influence of the SPH kernel.

Another boundary condition is the stress-free condition for momentum equation (1), which states that the total normal stress must be zero at free surface. Mathematically, it can be expressed as $(-p\mathbf{I} + \tau) \cdot \mathbf{n} = \mathbf{0}$, where $\mathbf{I}$ is the identity matrix and $\mathbf{n}$ is the normal to the free surface. This condition is trivially satisfied by the SPH gradients because the boundary integrals are ignored in the SPH derivatives approximation [26].

C. Implementation

We implement our technique in C++ and CUDA C++ using the Microsoft Visual Studio 2012 for Windows and CUDA 6.0. The input data of the program are given in an initialization file or interactively by the user and include: the material properties of the fluid model, configuration of the jet (shape, position, trajectory along the simulation, injection rate, maximum number of particles), and simulation parameters (time-step δt, maximum number of steps, total time), among others. For each time-step δt, the program updates the particle attributes following the sequence in Algorithm 1.

The first task is to add new fluid particles into the scene, which are generated by the jet inlet in CPU and then sent to a preallocated area (enough to store the maximum number of

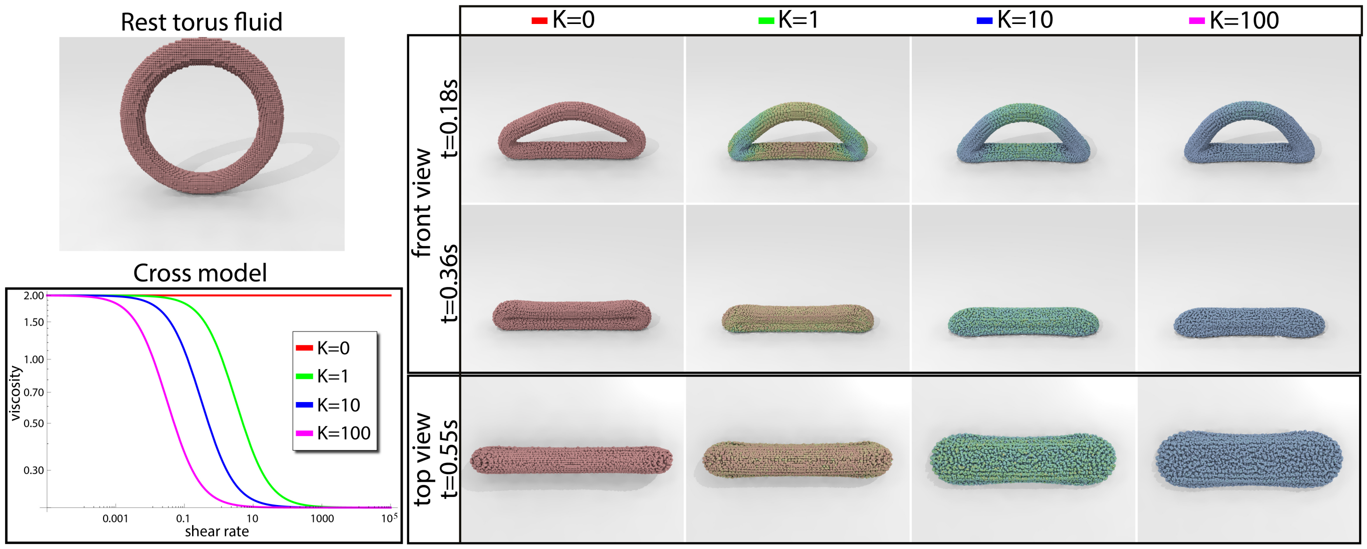


Fig. 3. Numerical simulation of the torus fluid spreading. The initial configuration of the torus fluid is illustrated in top-left. In bottom-left, we plot (loglog scales) the Cross model with parameters: $[\nu_\infty, \nu_0] = [0.2, 2]$, $n = 1$ and $K = 0,\ 1,\ 10$ and 100. Fluid flow simulations at different times (rows) in the right side: the first column illustrates a Newtonian fluid ($K = 0$) and non-Newtonian fluids on the remaining columns. The colors code the viscosity $\nu_i \in [\nu_\infty, \nu_0]$ in the SPH particles.

Algorithm 1 Single time step of the SPH framework

```
 1: Inject new fluid particles into the scene
 2: for each particle $i$ do
 3:     Find neighbors $N_i$
 4: end for
 5: for each particle $i$ do
 6:     Update $\rho_i$ using Equation (5)
 7:     Update $p_i$ using Equation (6)
 8: end for
 9: for each particle $i$ do
10:     Update $\nabla \mathbf{v}_i$ using Equation (8)
11:     Update $\tau_i$ using Equation (2)
12: end for
13: for each particle $i$ do
14:     Compute pressure acceleration using Equation (7)
15:     Compute viscous acceleration using Equation (9)
16: end for
17: for each particle $i$ do
18:     Update $\mathbf{v}_i$ and $\mathbf{x}_i$ with leap-frog scheme
19: end for
```

particles) in the global memory of the GPU. New particles are created iff:

1) The set of particles previously generated by the jet inlet has moved a given distance from the jet position;
2) The maximum number of particles into the scene has not yet been reached.

The data structure for the particle system representing the fluid is implemented using the Thrust library [27] and organized as a structure of arrays (of position, velocity, density, pressure, etc.) in order to allow coalesced access to the GPU global memory.

Next tasks are performed entirely in parallel on GPU. For the neighbor search, we use the algorithm described in the *Particles* demo in the CUDA's toolkit [28], which is based on a subdivision of the simulation space into a regular grid of linearly indexed cells. The algorithm relies on several CUDA kernels. The first one calculates a hash value for each particle based on the index of the cell containing the particle. We then use the Thrust sorting function to reorder the particles according to their hash values. Finally, a kernel which uses a thread per particle builds its neighbor list by comparing the cell index of the current particle with the cell index of the previous particle in the sorted list (see [28] for details).

The remaining tasks in Algorithm 1 are implemented by four CUDA kernels which also use one thread per particle. Each kernel computes, respectively, density and pressure, deformation tensor and shear stress, pressure and viscous accelerations, and velocity and position of each particle. We use the leap-frog scheme to integrate the system of differential equations provided by the SPH approximation of (1) along the particle trajectories. The numerical stability in this explicit time integration scheme is ensured under time-step constraints given by Courant-Friedrichs-Lewy (CFL) and viscous force conditions [19]:

$$\delta t \leq 0.1 \min\left(\frac{h}{c}, \frac{h^2}{8\nu_0}\right). \tag{10}$$

Respecting the Equation (10), we choose $\delta t = 10^{-6}$ seconds in our experiments.

The free surface rendering is performed by blobs (metaballs) from POV-Ray. The blob radius is equal to the radius of influence of the SPH kernel used in the simulation.

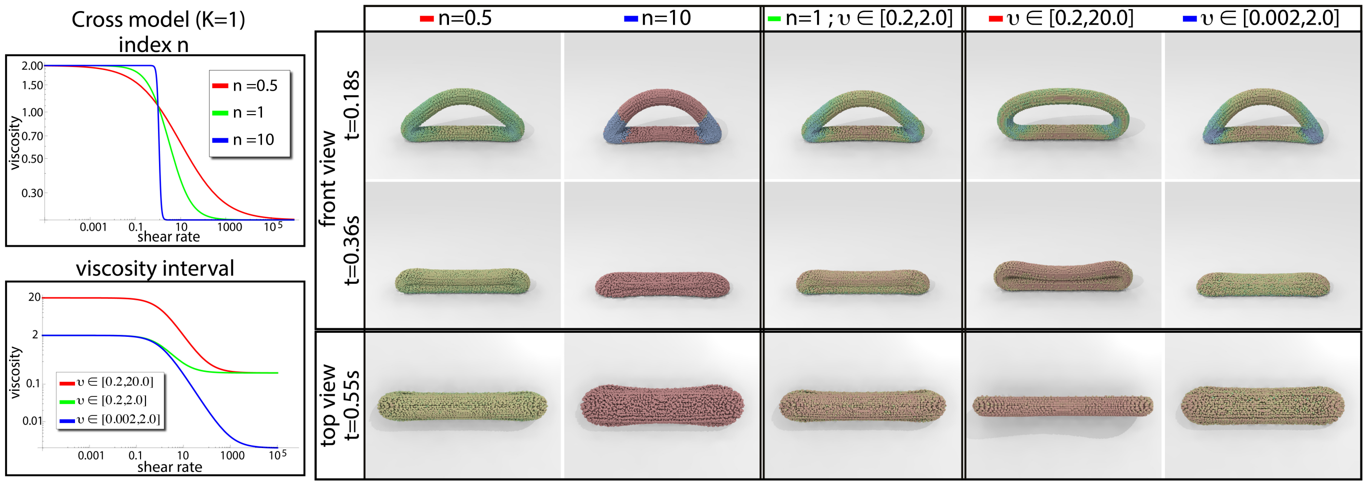


Fig. 4. Numerical simulation of the torus fluid spreading (Figure 3 shows initial configuration of the torus fluid). Fixing $K = 1$, the graphs (log-log scales) illustrate the Cross model with parameters: $[\nu_\infty, \nu_0] = [0.2, 2]$, $n = 0.5,\ 1.0$ and 10 (top-left); and $n = 1$, $[\nu_\infty, \nu_0] = [0.2, 2],\ [0.2, 20]$ and $[0.002, 2]$ (bottom-left). In the right side, the respective simulations at different times (rows) are shown. The first three columns illustrate the non-Newtonian fluid with variations of the parameter n. The last three columns, non-Newtonian fluid with distinct viscosity ranges. The colors code the viscosity $\nu_i \in [\nu_\infty, \nu_0]$ in the SPH particles.

IV. Results and Comparisons

In this section, we present the results provided by our technique, including animations of viscous liquids with coiling and folding effects.

Initially, we discuss the variable viscosity effects of non-Newtonian fluids in simulations of gravity spreading. Then, we apply our technique to a variety of situations involving jet buckling.

All examples have been generated in a computer equipped with a CPU Intel i5 3570 3.4Hz with 8GB of RAM, and a NVIDIA GeForce 650 with 384 CUDA cores and 1GB of RAM.

Torus fluid

In this first result, to demonstrate that our method can model shear-thinning behavior using the Cross model, we performed the numerical simulation of a torus fluid spreading on a plane surface due to gravity ($9.8\,m/s^2$).

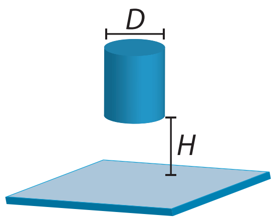
Figure 3 shows the results of our method using the Cross model with $n = 1$, $\nu_0 = 2.0$, $\nu_\infty = 0.2$ and four values of K, namely $K = 0$ for a Newtonian fluid (which corresponds to assign a constant viscosity at ν_0) and $K = 1.0,\ 10.0$ and 100.0 for a non-Newtonian fluid. The shear-thinning is increased at higher values of K allowing the fluid to spread more over the surface (see last row).

Another feature of the Cross model is the parameter n which determines the rate of decay of the viscosity as a function of the shear rate. The viscosity transition between the viscosity limits ν_∞ and ν_0 becomes faster at high values of n. Moreover, different rates between these limits also produce different behaviors in the fluid flow.

Figure 4 illustrates the effect of the parameters n, ν_∞ and ν_0 in our method, fixing $K = 1$. Left graphs show the viscosity variation under influence of those parameters, varying either n (top) or $[\nu_\infty, \nu_0]$ (bottom). On the right, corresponding simulations with those combinations of parameters are shown. The first three columns show the viscosity function, setting $K = 1$ and $[\nu_\infty, \nu_0] = [0.2, 2]$, varying the index parameters $n = 0.5,\ 1$ and 10. The last three columns show the viscosity function depending only of ν_∞ and ν_0. Fixing $n = 1$, we test different ranges of viscosity $[\nu_\infty, \nu_0]$: $[0.2, 2]$, $[0.002, 2]$ and $[0.2, 20]$.

Figures 3 and 4 help us to understand the shear-tinning behavior provided by Cross model with different sets of parameters. The results show that the Cross model determines the spatial variation of the viscous force that reflects the behavior of the fluid flow. This model allows a wide range of viscous effects of the free surface of Newtonian and non-Newtonian fluids in a small set of user parameters.

Jet buckling



The Jet buckling problem has been studied in several experimental and analytical investigations [29], [30]. This phenomenon is characterized by the formation of a physical instability when a thin viscous jet hits a rigid plate. The geometry of a jet buckling problem is given by the height H between the inlet and the rigid plate and by the jet width D in meters. In our simulations of this problem, we use values for H, D and physical parameters consistent with the conditions described by Tomé et al. [13].

When a stream of viscous fluid falls onto a surface, the deceleration of the stream near impact causes the buckling effect and the fluid starts to coil on itself. Figure 5 illustrates the coiling formation in our results. Note that, as the coiling

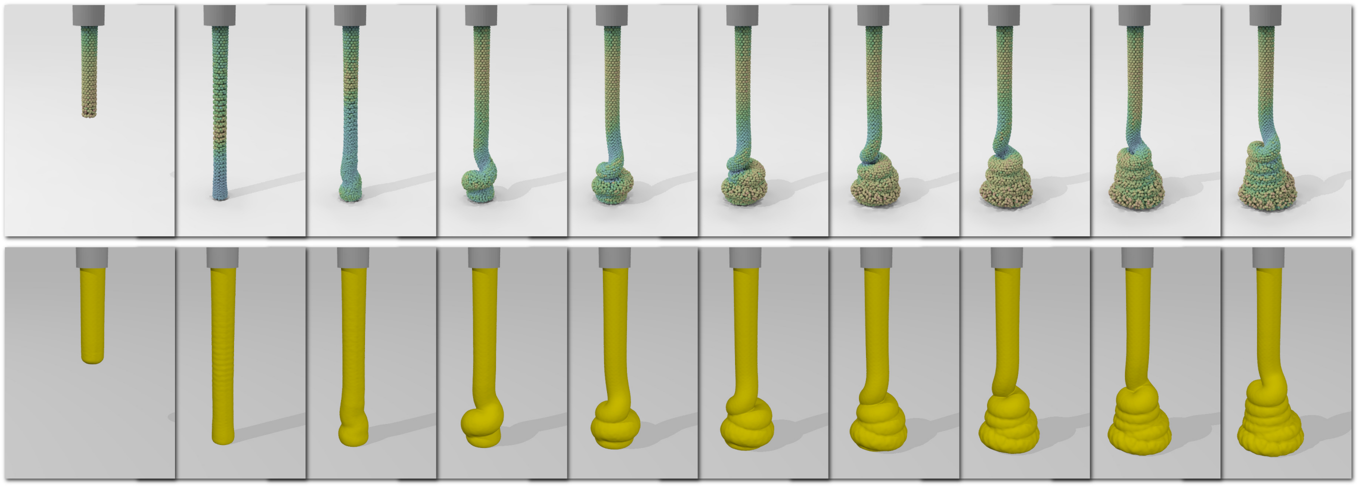


Fig. 5. The formation of coils in a falling stream of viscous liquid.

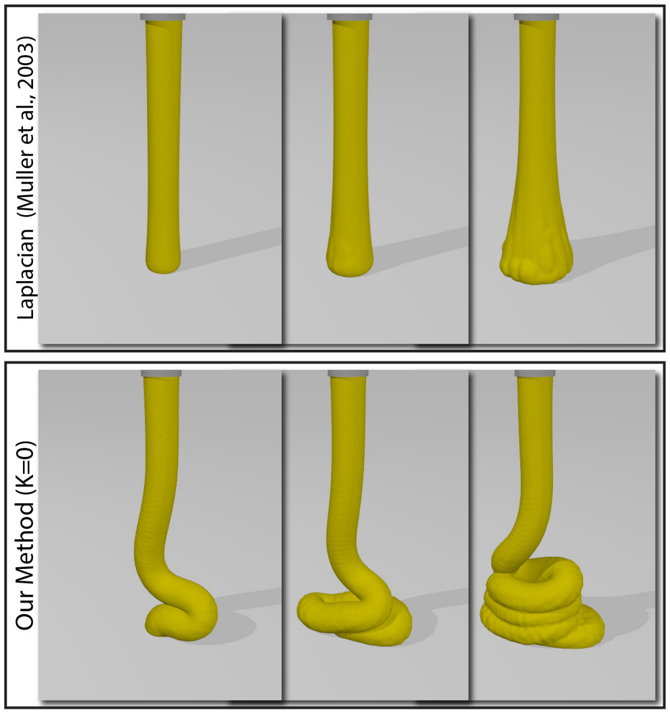


Fig. 6. Jet buckling of a Newtonian fluid. In top row, the result given by an implementation of [23] which uses the Laplacian velocity operator in the viscous acceleration. In bottom row, we show the result achieved by our technique with same parameters (Cross model with $K = 0$).

develops, the fluid tends to have a more viscous behavior at the surface, which prevents the fluid from further spreading. Hence, when the falling stream contacts the coils already formed, the viscosity decreases and new coils are created.

Figure 6 compares our technique with the popular SPH framework proposed by Müller et al. [23] in which the viscous term of Equation (1) is simplified using the Laplacian velocity operator $\nu_0 \Delta \mathbf{v}$. Choosing the same parameters in both simulations, we can verify that using the Laplacian velocity prevents the buckling development, leading to a spurious result.

In order to illustrate the relevance of the viscosity variation in our method, Figure 7 shows two jet buckling with distinct behaviors by applying Cross model in different viscosity intervals. Despite the fact that the fluids are fairly viscous, we can note, on the bottom of the fluid, the shear-thinning action, where the fluid becomes less viscous, spreading and mixing.

The shear-thinning behavior provided by inlet flow using rectangular jets are illustrated in Figure 8. The flow behavior of the Newtonian fluid (using $K = 0$ in Cross model) is more viscous than non-Newtonian fluid (using $K = 1$) due to the viscosity variation given by our technique. It is noted that the number of layers is reasonably pronounced in the Newtonian case and less pronounced in the non-Newtonian case.

Moving inlet jet

Figure 9 illustrates some ‘stitching’ patterns obtained with our method when a viscous liquid is injected by a moving jet. The buckling of the fluid and motion of jet combine to give a wide range of regular and periodic patterns, like a “sewing machine”.

Comparison between CPU and GPU

Table I compares the performance of our implementation running a dam break sequentially on CPU and in parallel on GPU. The parameters of the Cross model are: $K = 1$, $n = 1$, $\nu_0 = 0.05$, and $\nu_\infty = 0.005$. The first column is the number of particles; the columns labeled CPU and GPU show the simulation times, in seconds, for 1000 steps on CPU and GPU, respectively; the column Speedup is the relation CPU/GPU; and the column Efficiency is the speedup divided by the number of CUDA cores used in the experiment (384). Though the low efficiency, the speedup of about 27 for de number of particles varying between 64K and 256K justifies the use of the GPU: keeping that speedup, the CPU time for the simulation of 1M particles (not measured in our test) would be about ∽3.3 hours, against ∽7.4 minutes on GPU.

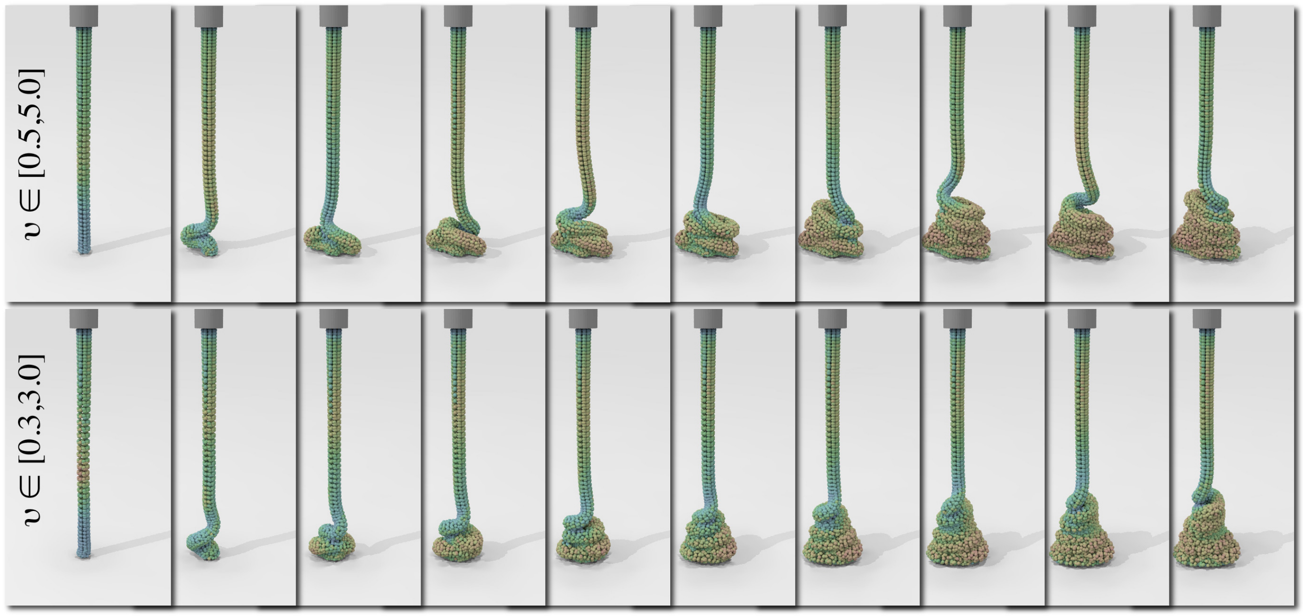


Fig. 7. Jet buckling with different viscosity intervals. Our method provides different coiling effects varying the parameters of Cross model.

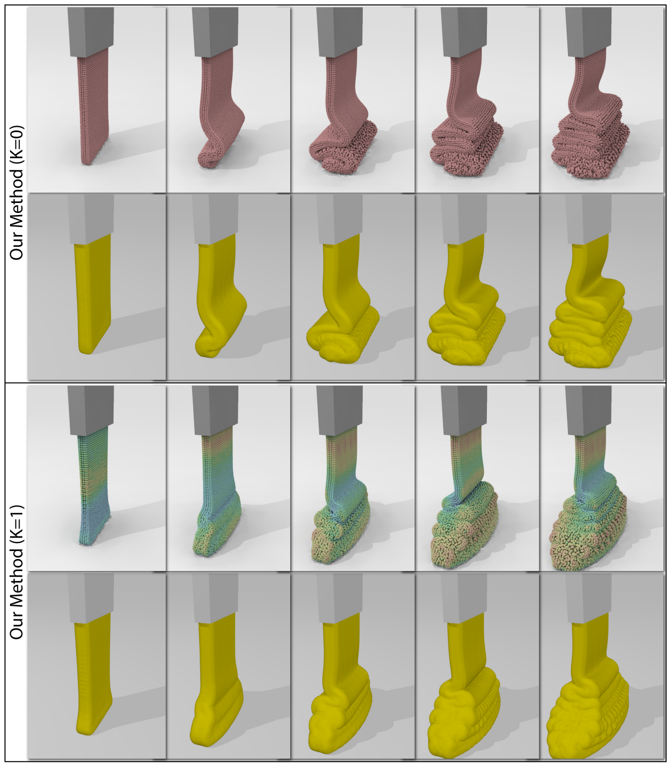


Fig. 8. Effect of the variable viscosity in our technique. At top, the simulation of a Newtonian fluid ($K = 0$) defines well-defined fluid layers, while a non-Newtonian fluid ($K = 1$) the fluid layers are mixed due to the shear-thinning.

V. Concluding remarks

In this paper, we introduced a novel SPH-based technique for animating free surface viscous liquids with jet buckling. Unlike the previous SPH frameworks, our technique allows visually realistic viscous behaviors, such as coiling and folding. The technique relies on the SPH approximation of a non-Newtonian fluid, where the variable viscosity is ruled by the Cross model. The effectiveness of the technique is demonstrated on simple and intuitive examples which match with the physics, leading to an efficient and attractive scheme for animation. Moreover, simulation time is considerably improved by using CUDA computing platform.

TABLE I
Performance Statistics.

# part.	**CPU**	**GPU**	**Speedup**	**Efficiency**
1K	$7.20s$	$1.03s$	7.0	1.8%
4K	$35.66s$	$2.40s$	14.9	3.9%
16K	$156.42s$	$6.81s$	23.0	6.0%
64K	$682.00s$	$25.80s$	26.4	6.9%
128K	$1374.86s$	$51.35s$	26.8	7.0%
256K	$2887.20s$	$104.84s$	27.6	7.2%
1M	—	$444.93s$	—	—

One limitation of our technique is that its results depend of the time-step size, the condition (10) may not permit large time-steps in the simulation, thus resulting a time consuming animation.

A natural direction for future work is to extend our technique to deal with truly incompressible fluid flows.

Acknowledgment

This work is supported by FAPESP (São Paulo Research Foundation) under grants #2013/19760-5 and #2014/11981-5, FAPES (Espírito Santo Research Foundation) grant #53600100/11 and CNPq (Brazilian National Council for Scientific and Technological Development). We would like to thank the anonymous reviewers for their valuable comments and suggestions. The real image of honey (Figure 1) was provided by Domiriel under Creative Commons license (`www.flickr.com/photos/domiriel/8037182858/`).

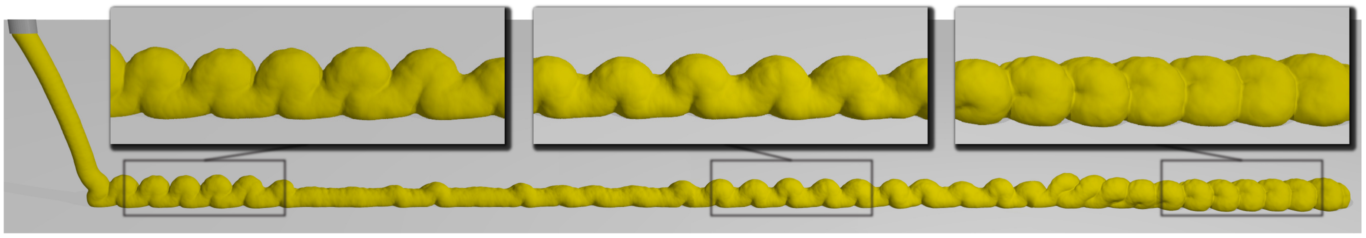


Fig. 9. Stitching patterns of a viscous thread poured by a moving jet.

References

[1] G. R. Liu and M. B. Liu, *Smoothed Particle Hydrodynamics*. World Science, 2005.

[2] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH Fluids in Computer Graphics," in *Eurographics 2014 – State of the Art Reports*.

[3] A. Herault, G. Bilotta, and R. A. Dalrymple, "SPH on GPU with CUDA," *Journal of Hydraulic Research*, vol. 48, no. Extra Issue, pp. 74–79, 2010.

[4] Ø. E. Krog and A. C. Elster, "Fast gpu-based fluid simulations using sph," in *Applied Parallel and Scientific Computing*. Springer, 2012, pp. 98–109.

[5] M. M. Cross, "Rheology of non-newtonian fluids: A new flow equation for pseudoplastic systems," *Journal of Colloid Science*, vol. 20, no. 5, pp. 417 – 437, 1965.

[6] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien, "A method for animating viscoelastic fluids," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 463–468, 2004.

[7] C. Batty and R. Bridson, "Accurate viscous free surfaces for buckling, coiling, and rotating liquids," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008, pp. 219–228.

[8] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, and E. Grinspun, "Discrete viscous threads," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 116:1–116:10, 2010.

[9] C. Batty, A. Uribe, B. Audoly, and E. Grinspun, "Discrete viscous sheets," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 113:1–113:7, 2012.

[10] C. Batty and B. Houston, "A simple finite volume method for adaptive viscous liquids," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11, 2011, pp. 111–118.

[11] M. F. Tomé and S. Mckee, "Numerical simulation of viscous flow: buckling of planar jets," *International Journal for Numerical Methods in Fluids*, vol. 29, no. 6, pp. 705–718, 1999.

[12] M. Tomé, A. Filho, J. Cuminato, N. Mangiavacchi, and S. Mckee, "GENSMAC3D: a numerical method for solving unsteady three-dimensional free surface flows," *International Journal for Numerical Methods in Fluids*, vol. 37, no. 7, pp. 747–796, 2001.

[13] M. Tomé, L. Grossi, A. Castelo, J. Cuminato, N. Mangiavacchi, V. Ferreira, F. de Sousa, and S. McKee, "A numerical method for solving three-dimensional generalized Newtonian free surface flows," *Journal of Non-Newtonian Fluid Mechanics*, vol. 123, no. 2–3, pp. 85 – 103, 2004.

[14] C. M. Oishi, M. F. Tomé, J. A. Cuminato, and S. McKee, "An implicit technique for solving 3d low Reynolds number moving free surface flows," *Journal of Computational Physics*, vol. 227, no. 16, pp. 7446–7468, 2008.

[15] S. Clavet, P. Beaudoin, and P. Poulin, "Particle-based viscoelastic fluid simulation," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005, pp. 219–228.

[16] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutré, and M. Gross, "A unified lagrangian approach to solid-fluid animation," in *Symposium on Point-Based Graphics 2005*, 2005, pp. 125–134.

[17] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares, "Particle-based non-newtonian fluid animation for melting objects," in *Sibgrapi 2006 (XIX Brazilian Symposium on Computer Graphics and Image Processing)*. IEEE, 2006, pp. 78–85.

[18] B. Solenthaler, J. Schläfli, and R. Pajarola, "A unified particle model for fluid-solid interactions," *Computer Animation and Virtual Worlds*, vol. 18, no. 1, pp. 69–82, 2007.

[19] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares, "Particle-based viscoplastic fluid/solid simulation," *Computer-Aided Design*, vol. 41, no. 4, pp. 306–314, 2009.

[20] A. Rafiee, M. Manzari, and M. Hosseini, "An incompressible SPH method for simulation of unsteady viscoelastic free-surface flows," *International Journal of Non-Linear Mechanics*, vol. 42, no. 10, pp. 1210 – 1223, 2007.

[21] X. Xu, J. Ouyang, B. Yang, and Z. Liu, "SPH simulations of three-dimensional non-Newtonian free surface flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 256, pp. 101 – 116, 2013.

[22] X. Xu and J. Ouyang, "A SPH-based particle method for simulating 3D transient free surface flows of branched polymer melts," *Journal of Non-Newtonian Fluid Mechanics*, vol. 202, pp. 54 – 71, 2013.

[23] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, pp. 154–159.

[24] J. P. Morris, P. J. Fox, and Y. Zhu, "Modeling low reynolds number for incompressible flows using SPH," *Journal of Computational Physics*, vol. 136, pp. 214–226, 1997.

[25] S. Koshizuka, A. Nobe, and Y. Oka, "Numerical analysis of breaking waves using the moving particle semi-implicit method," *International Journal for Numerical Methods in Fluids*, vol. 26, no. 7, pp. 751–769, 1998.

[26] J. Fang, R. G. Owens, L. Tacher, and A. Parriaux, "A numerical study of the SPH method for simulating transient viscoelastic free surface flows," *Journal of Non-Newtonian Fluid Mechanics*, vol. 139, no. 1–2, pp. 68–84, 2006.

[27] J. Hoberock and N. Bell, "Thrust: A parallel template library," 2010, version 1.7.0. [Online]. Available: http://thrust.github.io/

[28] S. Green, "Particle simulation using CUDA," 2012. [Online]. Available: docs.nvidia.com/cuda/samples/5_Simulations/particles/doc/particles.pdf

[29] J. O. Cruickshank, "Low-reynolds-number instabilities in stagnating jet flows," *Journal of Fluid Mechanics*, vol. 193, pp. 111–127, 1988.

[30] M. J. Blount and J. R. Lister, "The asymptotic structure of a slender dragged viscous thread," *Journal of Fluid Mechanics*, vol. 674, pp. 489–521, 2011.