DOI: 10.1111/cgf.13907
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Asynchronous Eulerian Liquid Simulation

T. Koike[†][1] S. Morishima[1][‡] and R. Ando[2][§]

[1]Waseda University, Japan
[2]National Institute of Informatics, Japan

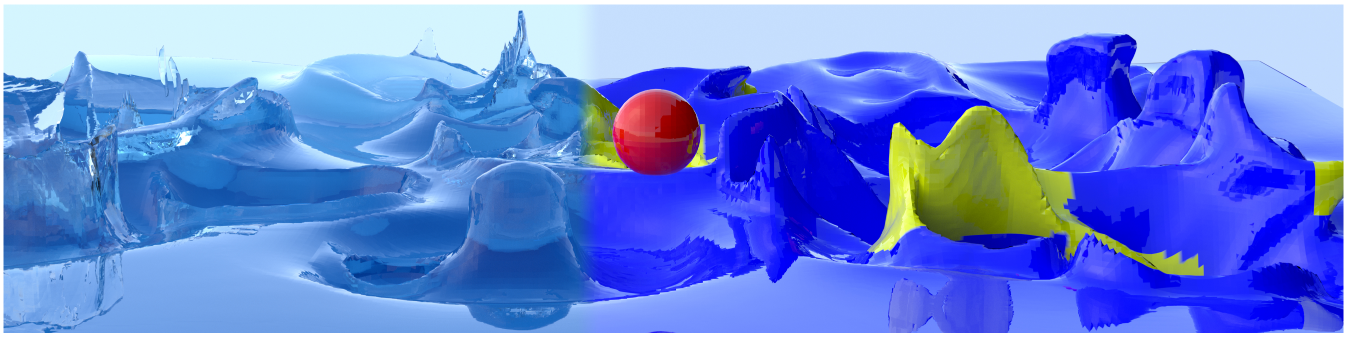

Figure 1: *Pouring rain: transparent view (left) and the sub-domain visualization (right). The peak of the speed-up ratio on this scene was* 7.20*, while the average was* 6.46*.* $384 \times 192 \times 384$ *grid resolutions. One video frame corresponds to 1/450 seconds in the real time scale. Compute time:* 52.26 *seconds per frame. Red color indicates the time step computed by the global time-stepping method. Yellow and blue colors indicate regions with three and nine times longer time step sizes, respectively.*

Abstract

We present a novel method for simulating liquid with asynchronous time steps on Eulerian grids. Previous approaches focus on Smoothed Particle Hydrodynamics (SPH), Material Point Method (MPM) or tetrahedral Finite Element Method (FEM) but the method for simulating liquid purely on Eulerian grids have not yet been investigated. We address several challenges specifically arising from the Eulerian asynchronous time integrator such as regional pressure solve, asynchronous advection, interpolation, regional volume preservation, and dedicated segregation of the simulation domain according to the liquid velocity. We demonstrate our method on top of staggered grids combined with the level set method and the semi-Lagrangian scheme. We run several examples and show that our method considerably outperforms the global adaptive time step method with respect to the computational runtime on scenes where a large variance of velocity is present.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Physical simulation;***

1. Introduction

To date, Eulerian liquid simulation has become a popular technique for simulating liquids in computer graphics. One of the advantages of Eulerian grids over Lagrangian methods such as SPH is the unconditional stability regardless of time step sizes. Such nature may tempt users to choose large time steps in the aim of reducing the whole runtime of simulation; however, such attempts can easily degrade the visual quality of small features such as turbulence and splashes due to the excessive violation of the Courant-Friedrichs-Lewy (CFL) condition, which means that a liquid should not travel more than the size of a cell within a single step [Bri08]. The CFL condition is often referred to as a stability condition if a finite-difference scheme is employed. Intuitively, a small bulk of liquid must collide multiple times with other parts of liquid in a long pe-

[†] mail: wassan2356@moegi.waseda.jp
[‡] mail: shigeo@waseda.jp
[§] mail: rand@nii.ac.jp

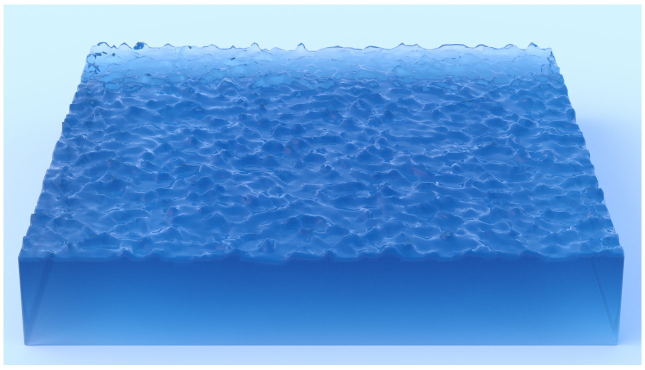

Figure 2: *Noisy perturbations on a static pool. A very large time-step ($\Delta t = 0.1$ on 128^3 resolutions) displays objectionable artifacts on the calm surfaces.*

riod of time but such a phenomenon is neglected when one chooses to advance time with a large time step.

We point out that using the method of Lentine et al. [LCPF12] may make large time steps doable, but issues explained above remain the same. Time steps with CFL numbers smaller than one would be ideal in this respect, but unfortunately, they introduce significant numerical diffusion. A similar issue is also reported by Setaluri et al. [SABS14] in their supplemental video. Empirically, we find that time steps around CFL $= 2$ are the sweet spot which provides qualitatively satisfactory results in practical scenarios when the semi-Lagrangian method is used for advection. Furthermore, our findings show that time steps with high CFL numbers induce noisy artifacts on calm surfaces as shown in Figure 2 when the operator splitting is employed for time integration. In this sense, feasible time step sizes for Eulerian methods are limited. This kind of problem is not discussed in previous work and we see that it is crucial for choosing adequate time step sizes without artifacts.

The state-of-the-art Eulerian liquid simulation employs the adaptive time step approach to determine the global time step size [Bri08]. The approach measures the maximal magnitude of velocity as a safe criterion to make sure that the CFL number at any location in space falls less than the target CFL number. This adaptive time-stepping works without excessive diffusion when the variance of velocity magnitude is small. However, when the variance is large such as a case where vibrant splashes are present, the CFL number of the majority part of liquid would undergo smaller than desired [FHHJ18], making the large portion of simulation unwantedly diffusive and inefficient.

In this paper, we propose a novel asynchronous time integrator for Eulerian liquid simulation. To the best of our knowledge, our work is the first to realize asynchronous liquid simulation on Eulerian grids in the context of computer graphics. The goal of our approach is to decompose the simulation domain into a few clusters and advance time with different time step sizes such that each next level of the cluster would have three times higher time step size, thereby achieving high efficiency in performance and minimizing excessive numerical diffusion due to the low CFL number.

The overview of our method is illustrated in Figure 3. Overall, our contributions are summarized as follows.

- Domain decomposition: we propose a new domain decomposition technique that ensures that all the local CFL numbers fall within the specified range.
- Both temporal and spatial interpolation of Eulerian materials: we present a new temporal-spatial interpolation scheme specifically tailored for our asynchronous integrator for level set, pressure, and velocity that is used to assign boundary values.
- Asynchronous advection scheme: we propose an accurate temporal advection scheme that is absent of visible seams on boundaries of regions of different time steps.
- Regional pressure solve: we solve for pressure supported by boundary pressure values interpolated from a region of different time steps. We also devise a new regional volume correction for our asynchronous method.

We compare with the reference solutions using the global adaptive time step algorithm [FF01; DG96] and show that our method does not introduce apparent visual compromise. Finally, we run several examples to demonstrate that our method can shorten the overall runtime.

2. Related Work

Advancing simulation with different time steps in part can be interpreted as "one-way" coupling in some specific scenario. Stomakhin et al. [SS17] proposed a flux animated boundary method where the analytical solution of a larger motion of liquid is used to specify the velocity flux on the boundary and is seamlessly blended with numerical simulations of liquid. Lentine et al. [LZF10] proposed a new multi-scale fluid simulation method where the pressure projection is performed in a coarse-to-fine fashion. Bojsen-Hansen and Wojtan [BW16] extended the Perfectly Matched Layer (PML) [SKM10] to allow a user to partially swap liquid simulations with new ones without artifacts on boundaries. Our method diverges from these methods in that we asynchronously integrate time.

Asynchronous methods have been applied to a wide variety of physical animations, including cloths [TPS08], contacts [HVS*12], liquids [OK12; GB14; RHEW17], and solids [GC01; SKZF11; ZLB16b; FHHJ18]. One of the challenges of asynchronous methods is retaining stability while minimizing artificial damping from large time steps; thereby combing explicit and implicit integrators [SKZF11]. We highlight that our method is not only unconditionally stable since we build our method on top of Fedkiw et al. [FF01] but also improves the artificial damping because we integrate time with adequate CFL numbers for each sub-domain. Another challenge is to optimize the decomposition of the simulation domain into sub-domains of different time steps. This is done by the use of velocity and force criterion [RHEW17] and block-wise criteria [GB14; FHHJ18]. Similar to the work of Goswami and Batty [GB14], our method only depends on velocity to determine the region of the time step. We show that our method is simple yet yields well-clustered sub-domains in practice.

Simulating fluid with large time steps can be an alternative way for speeding up the total simulation run-time. Lentine et

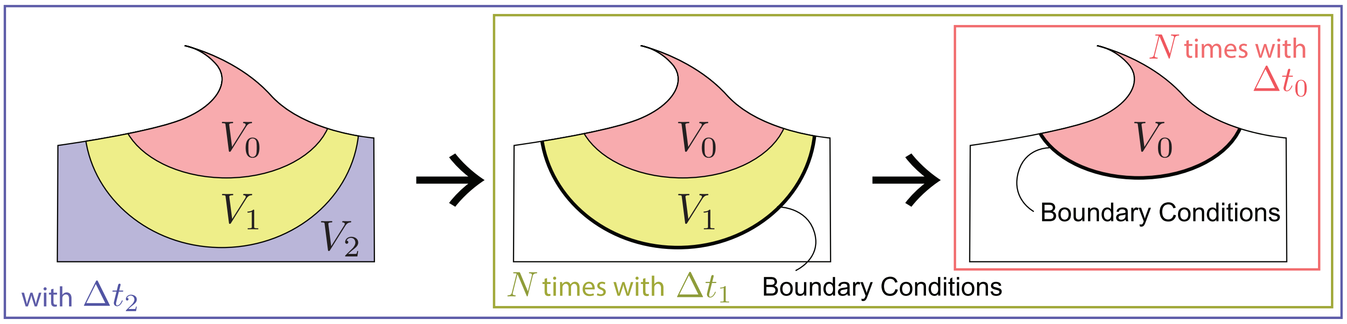

Figure 3: *Our method overview. We first partition the domain and assign different time step sizes for each of these. Next we advance the region $V_0 \cup V_1 \cup V_2$ (left). We use the results on the boundary between V_1 and V_2 as boundary conditions and re-simulate the region $V_0 \cup V_1$ using smaller time steps (middle). Similarly, we use the results on the boundary between V_0 and V_1 as boundary conditions to re-simulate V_0.*

al. [LCPF12] introduced a method that can take significant large time steps for liquid simulation through the use of a new advection scheme. We note that the purpose of their method is orthogonal to ours since they aim at enabling high CFL numbers while our method aims at eliminating extremely low CFL numbers that arise from global time steps where the largest velocity is used to determine the entire CFL; thereby, minimizing numerical dissipation. Hence, our method may be combined with the work of Lentine et al. [LCPF12] if this is desired.

We use a purely Eulerian method for simulating liquids, but FLIP(Fluid-Implicit-Particles) [ZB05] is in practice combined for production-quality visual simulation [Bri08]. In doing so, Nielsen and Bridson [NB11] proposed a two-pass simulation method where the first-pass simulation on a coarse grid is used to assign boundary conditions and re-simulate part of liquid with higher resolutions to enrich the motion near the surfaces without changing the overall motion. Zhang et al. [ZLB16a] proposed a one-way coupled adaptive pressure solver to resolve small-scale details for smoke simulation. Both methods utilize FLIP to express turbulence near solid boundaries or particulate splashes. While we believe that our method can be also combined with FLIP since our method can be regarded as an extension to Nielsen and Bridson [NB11], we leave such an extension for future work.

3. Method Overview

We base our method on Bridson [Bri08] to perform a liquid simulation. To do so, we solve for the Euler equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{g}, \qquad \nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

where $\boldsymbol{u}, t, \rho, p$ and $\boldsymbol{g}$ denote velocity, time, density, pressure and gravity, respectively. We enforce the Dirichlet boundary condition for free surfaces: $p = 0$ and the Neumann boundary condition on static solid interfaces: $\boldsymbol{n} \cdot \nabla p = 0$ where $\boldsymbol{n}$ denotes a solid normal. We discretize Eq.(1) over the staggered grid and apply the ghost fluid method on free surfaces [ENGF03]. To track the moving free surfaces, we use the level set method [OF01]. We integrate time using the operator splitting and the Euler method. Fortunately, unlike FEM, SPH or MPM, our explicit-time integration is unconditionally stable regardless of a time step size [Bri08] and as such stability is not a concern. Nevertheless, taking a large time step can degrade the visual quality as mentioned above.

3.1. Domain Decomposition and Time Step Estimation

We partition the simulation domain Ω into several clusters and assign adequate time step sizes such that all the local CFL numbers approach a desired target. Note that we refer "level" as a partitioned cluster hereafter. Here, time step sizes in each level k are denoted as Δt_k. Specifically, we assign Δt_k where $k > 0$ as follows:

$$\Delta t_k = N \Delta t_{k-1}, \tag{2}$$

where N denotes an interval coefficient which we used $N = 3$ in our examples. We compute Δt_0 according to the maximal velocity $\boldsymbol{u}_{\max}$ in the whole domain such that:

$$\Delta t_0 = \frac{c \Delta x}{||\boldsymbol{u}_{\max}||}, \tag{3}$$

where c and Δx denote a target CFL number coefficient and the grid cell size, respectively. We use $c = 2$ in our examples. However, we find that a very large time step induces instability as in Figure 2. To prevent the issue, we set the maximal time step $\Delta t_{\max}$ to $1/60$ in our examples. Note that we used the same Δt_0 as the time step size for the global time stepping method in the comparisons. The maximal velocity $\boldsymbol{u}_{\max}$ is then assigned as a representative velocity for the domain level zero as $\boldsymbol{u}_0 = \boldsymbol{u}_{\max}$. Accordingly, the rest of the representative velocity for the domain level $k > 0$ is given as:

$$\boldsymbol{u}_k = \frac{\boldsymbol{u}_{k-1}}{N}. \tag{4}$$

Along the way, we partition the domain Ω according to the formula:

$$V_k = \{\boldsymbol{x} \in \Omega \mid ||\boldsymbol{u}_{k+1}|| < ||\boldsymbol{u}(\boldsymbol{x})|| \leq ||\boldsymbol{u}_k||\}, \tag{5}$$

where V_k denotes a sub-domain of level k. We point out that our decomposition ensures that the local CFL number in every domain

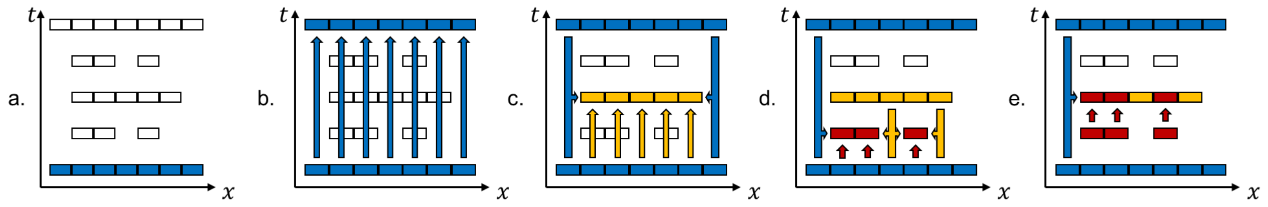

Figure 4: *1D example of our time integration overview with N being set to* 2*. Cell represents discretized variables (e.g., level set and velocity). Color refers to the level of regions (red: 0, yellow: 1, blue: 2). First, we advance the entire region with the largest time step* Δt_m *(b). Next, we advance a smaller time step* Δt_{m-1}*. The boundary values between the regions level m and* $m-1$ *are interpolated from the level m as illustrated with horizontal arrows (c). Subsequently, we advance smaller steps (d). Once we reach the smallest time step* Δt_0*, we advance the small time step again and overwrite variables of level* 1 *(e). We recursively apply these procedures to update/overwrite all the remaining variables.*

falls in the range:

$$c/N < \Delta t_k ||\boldsymbol{u}(\boldsymbol{x})||/\Delta x \leq c \quad \forall\, \boldsymbol{x} \in V_k. \tag{6}$$

In practice, this can generate numerous clusters if the minimal velocity is very small. To circumvent such issue, we set the maximal number of level m and set $\boldsymbol{u}_{m+1} = 0$. In our examples, we set $m = 2$. Note that m is at the same time the total count of levels minus one.

In addition to this, we apply the following extrapolation strategies to better minimize loss of visual features such as the sudden emergence of splashes. First, we place virtual particles $\boldsymbol{p}_0$ at the center of fluid cells. These particles are advected forward by velocity such that: $\boldsymbol{p}_1 = \boldsymbol{p}_0 + \Delta t_m \boldsymbol{u}_c$ where $\boldsymbol{u}_c$ denotes a cell velocity. Next, we rasterize all the cells between $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ and overwrite the level if the level on the cell is larger than the originating cell at $\boldsymbol{p}_0$. Finally, we extrapolate levels on every cell 6 cells wide towards larger levels. An overview of our extrapolation procedures is provided in the supplemental material.

3.2. Asynchronous Time Integration

Once the domain partition is complete, we advance simulation in the following order: from the largest level k, we concatenate all the sub-domains from the finest level to the level k such that $V_k^* = V_0 \cup V_1 \cup \cdots \cup V_k$, and advance the domain V_k^* using the time step Δt_k. In this way, a preliminary simulation state $Q_k(\boldsymbol{x} \in V_k^*, t + \Delta t_k)$ is known. Next, we decrement k and repeat the same procedure above. When computing pressure and advection on domain boundaries, we interpolate values (level set, velocity, and pressure) from a set of Q_{k+1} and specify as the Dirichlet boundary condition. The overview of our asynchronous time integration is illustrated in Figure 4.

Note that we do not impose Neumann boundary conditions on the pressure at domain boundaries like Nielsen et al. [NB11], but *fixed* boundary conditions (this is somewhat analogous to the free surface boundary conditions) which means that if the boundary pressure values are accurate, the solution is also accurate. Hence, even in the worst case that some (union) region does not touch air cells at all, our solver converges and gives some approximation. However, our boundary conditions come with slight divergence error as we discuss in more depth in Section 5.

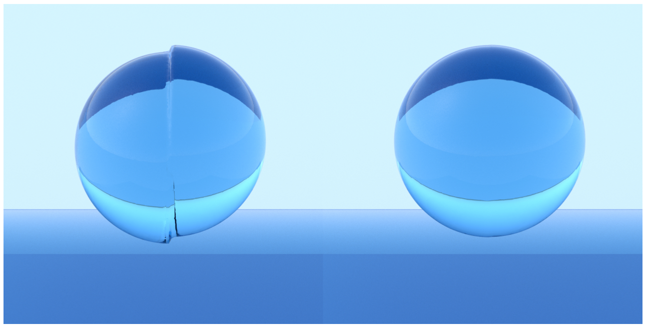

Figure 5: *Discrepancy issue on a free-falling sphere with a level transition at the center. Left: a naive asynchronous advection. Right: our modified scheme.*

As we detail in Section 3.4, a simple linear interpolation does not suffice and requires careful treatments. Finally, all Q_k are combined to compute the global physics state at the time $t + \Delta t_m$. Our asynchronous time step algorithm is inspired by the work of Fang et al. [FHHJ18] in the sense that we solve for the larger time steps first and use it as the boundary conditions to progressively solve for the remaining smaller time steps.

3.3. Asynchronous Advection

Naively advecting velocity and level set using different time steps can introduce visible discrepancy on the domain boundaries. Figure 5 left illustrates an example on a free-falling droplet across a domain transition. We clarify this issue with a simple example. Suppose that there is a single particle at a position $\boldsymbol{p}$ with zero velocity and let it freely fall by gravity. If we advance the particle state using the Euler method, the particle velocity at the next time step $t + \Delta T$ can be computed as $\Delta T \boldsymbol{g}$ while the particle position remains at $\boldsymbol{p}$. In contrast, if we use the half time step $\Delta t = \Delta T/2$ and advance twice, the particle velocity at $t + \Delta T$ will be the same as $2\Delta t \boldsymbol{g} = \Delta T \boldsymbol{g}$ but the particle position diverges: $\boldsymbol{p} + \Delta t^2 \boldsymbol{g}$. A similar mechanism is also present in the semi-Lagrangian advection scheme and this difference accumulates over time to develop objectionable seams.

We circumvent this issue by the following algorithm: before we advance a step, we extrapolate $\boldsymbol{a} = \boldsymbol{g} - \frac{1}{\rho}\nabla p$ using pressure computed from (multiple) larger levels. The extrapolation bandwidth depends on the target CFL number. Next, we perform a devised semi-Lagrangian advection:

$$q^*(\boldsymbol{x}, t+\Delta t) = q(\boldsymbol{x} - \Delta t\boldsymbol{u} - \int_0^{\Delta t} t\boldsymbol{a}(\boldsymbol{x})dt, t) \quad (7)$$

$$\approx q(\boldsymbol{x} - \Delta t\boldsymbol{u} - \Sigma_{n=1}^{T} n\Delta t_k^2 \boldsymbol{a}(\boldsymbol{x}), t), \quad (8)$$

where q denotes a quantity to advect such as velocity or level set, q^* denotes an intermediate quantity after the advection, and $T = \Delta t/\Delta t_k$, assuming that we solve for the level k. Note that in Eq.(8) we use a discrete approximation instead of the analytical integration because in this way advection on the level zero will not need our modification. Figure 5 right shows an example of our method applied to the same setup of Figure 5 left, which successfully eliminates the discrepancy issue.

3.4. Interpolation

When interpolating a value between time t and $t + \Delta t_k$, we use different strategies depending on what kind of quantity and what kind of attribute (space or time) to interpolate. First, we only consider the time attribute. Pressure computed on the state $Q_k(t + \Delta t_k)$ is interpreted as $p_k(t + \Delta t_k)$ such that:

$$p_k(t+\Delta t_k) = \frac{1}{\Delta t_k}\int_t^{t+\Delta t_k} p(t)\ dt, \quad (9)$$

which means that unlike velocity, our discrete pressure represents the *average* of pressure between two time steps, not exactly a quantity at a specific time. Therefore, we select $Q_k(t + \Delta t_k)$ for pressure interpolation for time between t and $t + \Delta t_k$. For velocity and level set, we select $Q_k(t)$ as follows.

We first extrapolate velocity at time t depending on the target CFL number c. For level set interpolation, we use the extrapolated velocity to back-trace the position using Eq.(8) and a target time interval Δt^*. The back-traced position is used to fetch the level set and pressure. Velocity is also similarly interpolated, but in the case of velocity, we add $\Delta t^* \boldsymbol{a}$ to consider the change of velocity due to the acceleration during the time integration. This interpolation scheme can be also used to interpolate fluid geometry at any timings we wish. For example, in Figure 1, level set of the blue region was interpolated twice (or sometimes three times) per (regional) time step (Δt_2) to generate multiple video frames.

3.5. Pressure Projection and Volume Correction

We perform pressure projection supported by the Dirichlet boundary conditions interpolated from the larger levels as described in Section 3.2. In this way, the pressure on the region boundaries is always given explicitly. Along the way, we correct the volume change (mostly the volume loss) introduced by the level set advection within this projection step extending the work of Kim et al. [KLL*07]. Let V be the current entire volume of a liquid, and V_{target} the target liquid volume. We first compute the volume change ratio: $\alpha = V_{\text{target}}/V$. Next, when performing pressure projection, we compute the union of sub-domains V_k^* as in Section 3.2 and define the target volume on this level as $V_{\text{target}}^k = \alpha \text{Vol}(V_k^*)$, where $\text{Vol}(\cdot)$ denotes a function to compute the volume. The remaining procedures for the volume correction are the same as the method of Kim et al. [KLL*07].

4. Results

Scene	Sec/Frame		Sec/Δt_m	Avg/Max Speed-up
	Sync	Async		
Fig.6	180.25	87.98	50.11	2.05 / 3.55
Fig.7	138.89	48.70	51.85	2.85 / 3.05
Fig.1	337.36	52.26	94.51	6.46 / 7.20

Table 1: *Timings and performance gain of our method. Sec/Δt_m denotes compute time to advance Δt_m.*

Table 1 illustrates the overview of timings and the performance gain of our method. Figure 8 shows a performance breakdown of our examples. Throughout the examples, we set the width of the domain one meter wide and gravity $9.8m/s^2$. All of our video examples run at 60FPS. We parallelized all of our examples where applicable using OpenMP. We used the semi-Lagrangian advection and the Incomplete Cholesky Preconditioned Conjugate Gradient method with the L_2 norm of 10^{-4} for pressure solve provided by Eigen library [GJ*10]. Instead of synchronizing time steps with the video frame rate [LCPF12], we chose to interpolate level set when the time exceeds the next video frame (as mentioned in Section 3.4). The supplemental video provides comparisons to all the corresponding reference solutions computed by the global time stepping method.

Figure 1 shows an example of our method on a pouring rain scenario. In this scene, small spherical liquids are injected from the top at a fast constant speed. In this specific scenario, the global time step method would compute a time step according to the velocity of fast-falling raindrops, making the time step size very small despite the majority part of liquid being rather calm. By using our asynchronous solver, the time step sizes are segregated according to the velocity magnitude, which resulted in up to 7.20× performance speed-up, and 6.46× on average. This example was ran on an Intel(R) Core(TM) i9-9980XE CPU @ 3.00GHz, 18 cores, 36 threads 128GB RAM machine. Regarding the volume correction, we set the target volume constant on this example to keep the water level static in the course of the simulation. The video was slowed down to 0.13× compared to the actual time speed and we used $384 \times 192 \times 384$ grid resolutions. We could not simulate thin sheets and particulate splashes on this example since we use a purely Eulerian level set method. Using FLIP may help reproduce such effects.

Figure 6 shows an example of a dam breaking demonstrating the general applicability of our method. Although the variance of velocity magnitude was comparably not large on this example, we were able to speed-up the simulation up to 3.55×, and 2.05× on average. This result indicates that our method would noticeably speed-up the average simulation time on many practical scenarios. This example was ran on an Intel(R) Core(TM) i7-6950X CPU @ 3.00GHz, 10 cores, 20 threads 128GB RAM machine. Volume fluctuation was less than 3%. The video plays at the speed of the actual

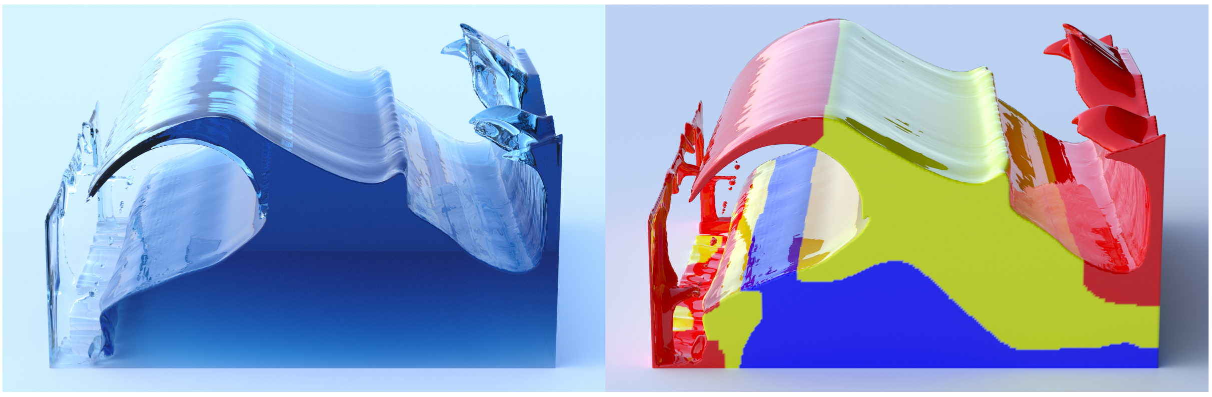

Figure 6: *Dam breaking.* 256^3 *resolutions. One video frame corresponds to 1/60 seconds in the real time scale. Red, yellow, and blue colors represent level 0, level 1, and level 2, respectively. Compute time:* 87.98 *seconds per frame. Speed-up gain was up to* 3.55 *and* 2.05 *on average. When the number of levels is only 2, yellow represents level 0 and blue 1.*

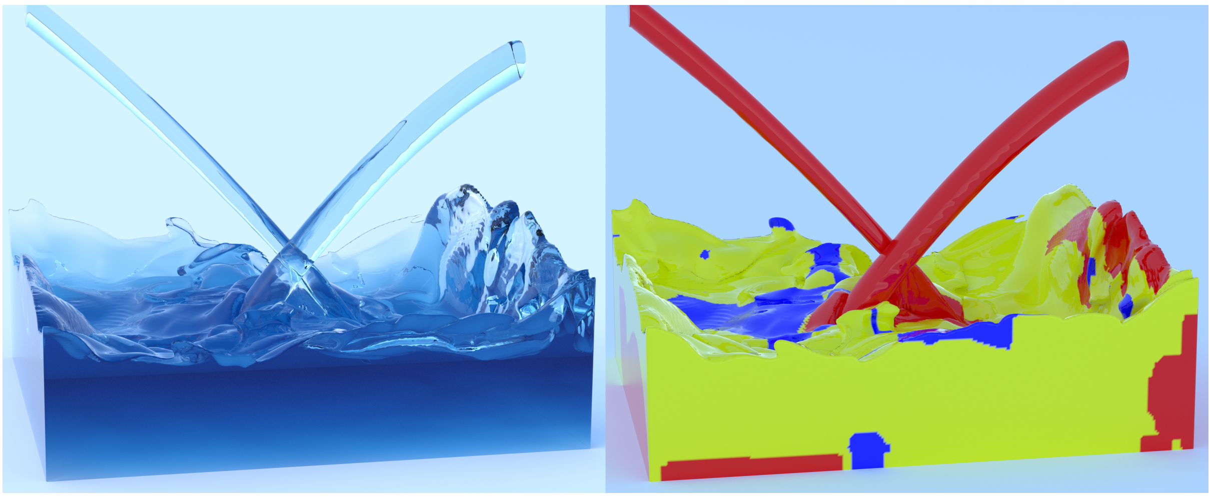

Figure 7: *Water spray.* 256^3 *resolutions. One video frame corresponds to 1/90 seconds in the real time scale. The color scheme is the same as the one in Figure 6. Compute time:* 48.70 *seconds per frame. Speed-up gain was up to* 3.05 *and* 2.85 *on average.*

time. We used the grid resolutions 256^3. The kinetic energy plot is shown in Figure 9, which indicates a reduced numerical diffusion.

Figure 7 shows an example of two columns of water spray injected from the left and right walls, colliding at the center. In this setup, the injected sprays are simulated with the smallest time step, and the remaining parts were simulated at larger time steps, resulting in up to $3.05\times$ faster performance gain and $2.85\times$ on average. We used the grid resolutions 256^3 and the video was slowed down to $2/3\times$ compared to the actual time speed. This example was ran on an Intel(R) Core(TM) i9-7920X CPU @ 2.90GHz, 12 cores, 24 threads 128GB machine.

5. Discussion and Limitations

Our domain decomposition does not consider temporal coherence. Nonetheless, we did not observe any severe temporal flickering on many of our examples. However, this does not guarantee that our method is flicker-free. We used the semi-Lagrangian method for level set advection because it is simple, fast and unconditionally stable; however, the semi-Lagrangian method may be too diffusive in practice [FF01]. As an alternative, one may extend our asynchronous scheme with higher-order methods such as a spline interpolation [FF01] and a MacCormack method [SFK*07]. We believe that this is possible; for example, a MacCormack method [SFK*07] can be seen as a two-pass semi-Lagrangian method in which each backward/forward tracing along the velocity field can be performed as in Section 3.3. We also tested a solid obstacle scene and it works without artifacts. No modification is needed except for velocity extrapolation into the solid, which is common for Eulerian liquids. More details are available in the supplemental material.

Throughout our examples, we used Fast Marching method [Bri08] due to its simplicity. Alternatively, a PDE-based approach [RS00] can be also employed, which we find provides smooth surfaces with reduced grid aligned artifacts. An

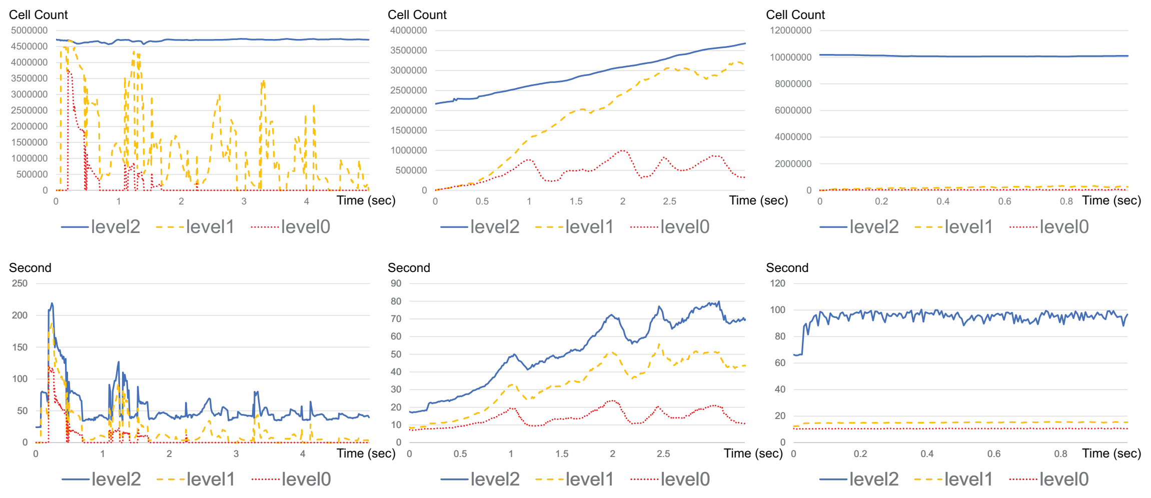

Figure 8: *From left to right, Fig.6, Fig.7, Fig.1. Top row: cell count. Bottom row: accumulated simulation time per level to advance Δt_m. Cell count indicates the total fluid cells in V_k^* as in Section 3.2. The compute time per level is inclusive: e.g., the blue line (level 2) includes timings for level 1 and 0. When the number of levels is only 2 (e.g., towards the end of Fig.6), yellow represents level 0 and blue 1.*

example is shown in the supplemental material and the video. In the specific setting of Figure 5, the motion of spherical liquid is simply ballistic, so that the contribution of pressure force does not exist. The only driving force is gravity, and the time integration error results in this discrepancy. Our new asynchronous advection in Section 3.3 fixes the issue.

We also explored a larger number on the maximal sub-domain level m, but we observed noisy perturbation artifacts as in Figure 2 on at least more than one of our examples (even though we set the maximal time step size $\Delta t_{\max}$). Extending the number m further requires research on perturbation condition and therefore is left for future work. We also observed that some sub-domains are topologically disconnected, which suggests that we can simulate them completely in parallel and gain further speed-up.

Our Dirichlet boundary conditions on pressure solve introduce some (slight) divergence near domain boundaries, which accumulates over time. For example, if we turn off the volume correction for Figure 6, we observed that the synchronous method loses 25.5% volume while ours 52.8% (2.07× volume loss). Note that if we use the Neumann boundary conditions as in the work of [NB11], we need to check if any (union) region is completely surrounded by Neumann boundaries by a flood-fill-like algorithm and modify the divergence, which would make the algorithm (overly) complex and slower on large-scales. We have tested a challenging case (water drop on a vertically long cubical tank with region horizontally split) and found that it works in practice, which is shown both in the supplemental material and the video. In this paper, we chose to focus on liquid first because the variance of velocity magnitude would be higher than smoke, and such a more speed-up gain was expected; however, we believe that our method can be also applied to smoke.

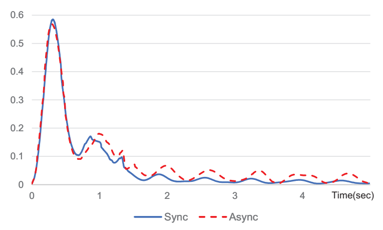

Figure 9: *Kinetic energy plot on Fig.6. The energy gradually settles both on asynchronous and synchronous methods, but our method better retains the energy bounce, indicating an improved numerical diffusion of velocity.*

6. Conclusion

We have presented an asynchronous time integrator for Eulerian liquid simulation. Our approach guarantees that the local CFL falls in the specified range, which results in considerable performance gain and the improved kinetic energy curve. Our method is straightforward to implement and can be expected to excel in performance on many practical scenarios. In the future, we would like to combine our method with Affine Particle in Cell (APIC) method [JSS*15] to enrich the detail such as splashes. We also would like to extend our method to smoke and handle two-way coupled rigid bodies.

Acknowledgements

This research is supported by the JST ACCEL Grant Number JPMJAC1602, JST-Mirai Program Grant Number JPMJMI19B2, JSPS KAKENHI Grant Number JP17H06101, JP19H01129 and 18K18060. We thank Kenshi Takayama for discussions regarding mathematical examination and proofreading the paper.

References

[Bri08] BRIDSON, ROBERT. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, Sept. 2008. ISBN: 1568813260 1–3, 6.

[BW16] BOJSEN-HANSEN, MORTEN and WOJTAN, CHRIS. "Generalized Non-reflecting Boundaries for Fluid Re-simulation". *ACM Trans. Graph.* 35.4 (July 2016), 96:1–96:7. ISSN: 0730-0301. DOI: 10.1145/2897824.2925963 2.

[DG96] DESBRUN, MATHIEU and GASCUEL, MARIE-PAULE. "Smoothed Particles: A new paradigm for animating highly deformable bodies". *Computer Animation and Simulation '96*. Ed. by BOULIC, RONAN and HÉGRON, GERARD. Vienna: Springer Vienna, 1996, 61–76. ISBN: 978-3-7091-7486-9 2.

[ENGF03] *Using the Particle Level Set Method and a Second Order Accurate Pressure Boundary Condition for Free Surface Flows*. Vol. Volume 2: Symposia, Parts A, B, and C. Fluids Engineering Division Summer Meeting. July 2003, 337–342. DOI: 10.1115/FEDSM2003-45144 3.

[FF01] FOSTER, NICK and FEDKIW, RONALD. "Practical Animation of Liquids". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 2001, 23–30. ISBN: 1-58113-374-X. DOI: 10.1145/383259.383261. URL: http://doi.acm.org/10.1145/383259.383261 2, 6.

[FHHJ18] FANG, YU, HU, YUANMING, HU, SHI-MIN, and JIANG, CHENFANFU. "A Temporally Adaptive Material Point Method with Regional Time Stepping". *Computer Graphics Forum* 37.8 (2018), 195–204. DOI: 10.1111/cgf.13524 2, 4.

[GB14] GOSWAMI, PRASHANT and BATTY, CHRISTOPHER. "Regional Time Stepping for SPH". *Eurographics 2014*. Ed. by GALIN, ERIC and WAND, MICHAEL. Short Papers. Strasbourg, France: Eurographics Association, Apr. 2014, 45–48. DOI: 10.2312/egsh.20141011. URL: https://hal.inria.fr/hal-00980592 2.

[GC01] GRAVOUIL, ANTHONY and COMBESCURE, ALAIN. "Multi-time-step explicit-implicit method for non-linear structural dynamics". *International Journal for Numerical Methods in Engineering* 50.1 (2001), 199–225. DOI: 10.1002/1097-0207(20010110)50:1<199::AID-NME132>3.0.CO;2-A 2.

[GJ*10] GUENNEBAUD, GAËL, JACOB, BENOÎT, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010 5.

[HVS*12] HARMON, DAVID, VOUGA, ETIENNE, SMITH, BREANNAN, et al. "Asynchronous Contact Mechanics". *Commun. ACM* 55.4 (Apr. 2012), 102–109. ISSN: 0001-0782. DOI: 10.1145/2133806.2133828 2.

[JSS*15] JIANG, CHENFANFU, SCHROEDER, CRAIG, SELLE, ANDREW, et al. "The Affine Particle-in-cell Method". *ACM Trans. Graph.* 34.4 (July 2015), 51:1–51:10. ISSN: 0730-0301. DOI: 10.1145/2766996. URL: http://doi.acm.org/10.1145/2766996 7.

[KLL*07] KIM, BYUNGMOON, LIU, YINGJIE, LLAMAS, IGNACIO, et al. "Simulation of Bubbles in Foam with the Volume Control Method". *ACM Trans. Graph.* 26.3 (July 2007). ISSN: 0730-0301. DOI: 10.1145/1276377.1276500. URL: http://doi.acm.org/10.1145/1276377.1276500 5.

[LCPF12] LENTINE, MICHAEL, CONG, MATTHEW, PATKAR, SAKET, and FEDKIW, RONALD. "Simulating Free Surface Flow with Very Large Time Steps". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '12. Lausanne, Switzerland: Eurographics Association, 2012, 107–116. ISBN: 978-3-905674-37-8. URL: http://dl.acm.org/citation.cfm?id=2422356.2422373 2, 3, 5.

[LZF10] LENTINE, MICHAEL, ZHENG, WEN, and FEDKIW, RONALD. "A Novel Algorithm for Incompressible Flow Using Only a Coarse Grid Projection". *ACM Trans. Graph.* 29.4 (July 2010). ISSN: 0730-0301. DOI: 10.1145/1778765.1778851. URL: https://doi.org/10.1145/1778765.1778851 2.

[NB11] NIELSEN, MICHAEL B. and BRIDSON, ROBERT. "Guide Shapes for High Resolution Naturalistic Liquid Simulation". *ACM Trans. Graph.* 30.4 (July 2011), 83:1–83:8. ISSN: 0730-0301. DOI: 10.1145/2010324.1964978 3, 7.

[OF01] OSHER, STANLEY and FEDKIW, RONALD P. "Level Set Methods: An Overview and Some Recent Results". *Journal of Computational Physics* 169.2 (2001), 463–502. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.2000.6636 3.

[OK12] ORTHMANN, JENS and KOLB, ANDREAS. "Temporal Blending for Adaptive SPH". *Comput. Graph. Forum* 31.8 (Dec. 2012), 2436–2449. ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.03186.x 2.

[RHEW17] REINHARDT, STEFAN, HUBER, MARKUS, EBERHARDT, BERNHARD, and WEISKOPF, DANIEL. "Fully Asynchronous SPH Simulation". *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '17. Los Angeles, California: ACM, 2017, 2:1–2:10. ISBN: 978-1-4503-5091-4. DOI: 10.1145/3099564.3099571 2.

[RS00] RUSSO, GIOVANNI and SMEREKA, PETER. "A Remark on Computing Distance Functions". *Journal of Computational Physics* 163.1 (2000), 51–67. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.2000.6553. URL: http://www.sciencedirect.com/science/article/pii/S0021999100965537 6.

[SABS14] SETALURI, RAJSEKHAR, AANJANEYA, MRIDUL, BAUER, SEAN, and SIFAKIS, EFTYCHIOS. "SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation". *ACM Trans. Graph.* 33.6 (Nov. 2014), 205:1–205:12. ISSN: 0730-0301. DOI: 10.1145/2661229.2661269. URL: http://doi.acm.org/10.1145/2661229.2661269 2.

[SFK*07] SELLE, ANDREW, FEDKIW, RONALD, KIM, BYUNGMOON, et al. "An Unconditionally Stable MacCormack Method". *Journal of Scientific Computing* 35.2-3 (Nov. 2007), 350–371. DOI: 10.1007/s10915-007-9166-4. URL: https://doi.org/10.1007/s10915-007-9166-4 6.

[SKM10] SÖDERSTRÖM, ANDREAS, KARLSSON, MATTS, and MUSETH, KEN. "A PML-based Nonreflective Boundary for Free Surface Fluid Animation". *ACM Trans. Graph.* 29.5 (Nov. 2010), 136:1–136:17. ISSN: 0730-0301. DOI: 10.1145/1857907.1857912 2.

[SKZF11] SCHROEDER, CRAIG, KWATRA, NIPUN, ZHENG, WEN, and FEDKIW, RON. "Asynchronous Evolution for Fully-Implicit and Semi-Implicit Time Integration". *Computer Graphics Forum* 30.7 (2011), 1983–1992. DOI: 10.1111/j.1467-8659.2011.02046.x 2.

[SS17] STOMAKHIN, ALEXEY and SELLE, ANDREW. "Fluxed Animated Boundary Method". *ACM Trans. Graph.* 36.4 (July 2017), 68:1–68:8. ISSN: 0730-0301. DOI: 10.1145/3072959.3073597 2.

[TPS08] THOMASZEWSKI, BERNHARD, PABST, SIMON, and STRASSER, WOLFGANG. "Asynchronous Cloth Simulation". 2008 2.

[ZB05] ZHU, YONGNING and BRIDSON, ROBERT. "Animating Sand As a Fluid". *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: ACM, 2005, 965–972. DOI: 10.1145/1186822.1073298. URL: http://doi.acm.org/10.1145/1186822.1073298 3.

[ZLB16a] ZHANG, XINXIN, LI, MINCHEN, and BRIDSON, ROBERT. "Resolving Fluid Boundary Layers with Particle Strength Exchange and Weak Adaptivity". *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925910. URL: https://doi.org/10.1145/2897824.2925910 3.

[ZLB16b] ZHAO, DANYONG, LI, YIJING, and BARBIČ, JERNEJ. "Asynchronous Implicit Backward Euler Integration". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '16. Zurich, Switzerland: Eurographics Association, 2016, 1–9. ISBN: 978-3-905674-61-3 2.