scientific reports

Check for updates

OPEN

Unified GPU-based simulation of porous flow, absorption, and diffusion in cloth–liquid coupling

Jong-Hyun Kim[1] & Jung Lee[2]✉

Cloth–liquid interaction involves coupled phenomena such as porous flow, absorption, emission, and diffusion, which are difficult to model in a stable and efficient manner in particle-based simulations. In this paper, we present a GPU-based cloth–liquid coupling framework that integrates these processes within a unified Smoothed Particle Hydrodynamics (SPH) pipeline. To improve directional robustness in porous flow near boundaries, we use a virtual-pressure formulation inspired by Darcy's law, while absorption and emission are handled through saturation-based mass exchange and diffusion is applied as a sequential redistribution stage. For efficient neighbor search, the framework employs a Bitonic sort–based GPU hashing structure. We evaluate the proposed method using qualitative wet-cloth scenarios and quantitative analyses including frame-time measurement, sorting-cost comparison, scalability with increasing particle counts, mass conservation error, and module-wise timing breakdown. The results show that the proposed hashing scheme reduces sorting overhead compared with the baseline GPU implementation, while the full coupling pipeline maintains stable wetting behavior and bounded mass variation across the tested scenes. These results indicate that the proposed framework provides an efficient and numerically stable approach for GPU-based porous cloth–liquid simulation.

Cloth-fluid interaction is one of the physical phenomena that can be easily observed in real life. Wet cloth exhibits a variety of complex physical phenomena. As water permeates the cloth, droplets can form as a result of surface tension, and the weight of the cloth increases as it absorbs water. Furthermore, the absorption of water can change the texture of the cloth or darken its color[1,2]. To implement these phenomena, it is essential to understand the composition and characteristics of the cloth or fabric. The fabric that makes up the cloth is composed of individual strands made up of thin fibers, and the liquid is absorbed both within these fibers and between them.

According to mixture theory, fabric can be modeled as a continuous porous medium through which fluid can flow[3]. This model explains the anisotropic structure of the material and changes in saturation and can represent buoyancy, drag, surface tension effects, and fluid convection. In this paper, we propose an efficient framework for representing interactions by integrating a mass-spring structured cloth simulation with SPH-based fluid simulation in a GPU environment. Existing techniques could only model specific phenomena, but the method proposed in this paper is an integrated simulation technique that can implement various physical properties that appear during fluid and cloth interactions within a single GPU framework. This allows users to quickly model and create scenes.

Each solver for physics-based simulations involves significant computational effort, and performing two-way coupling of two or more materials requires even more computation. NVIDIA's CUDA programming is a representative architecture for parallelizing and optimizing computationally intensive problems, and is widely used in fields such as geometry, simulation, and numerical analysis[4–6]. Particle simulation, in particular, requires an efficient data structure as the number of particles increases. In CUDA-based particle simulation provided by NVIDIA, performance is enhanced using the Thrust parallel algorithm library with a CUDA-based hash table. Although this method shows fast simulation performance with a large number of particles and is utilized in various techniques, there is an issue of reduced computational performance during the data sorting phase. In the coupling algorithm addressed in this paper, the problem becomes more severe as the number of nodes and particles increases. Fig 1 shows the computation time for each function in the hash table presented by CUDA. As shown in the figure, most of the computation time is spent on the sorting process. This paper proposes a new

[1]College of Software and Convergence (Department of Artificial Intelligence, Design Technology), Graduate School of Electrical and Computer Engineering, Inha University, Incheon 22212, Michuhol-gu, South Korea. [2]Department of Computer Engineering, Hanbat National University, Daejeon 34158, Yuseong-gu, South Korea. ✉email: airjung@hanbat.ac.kr

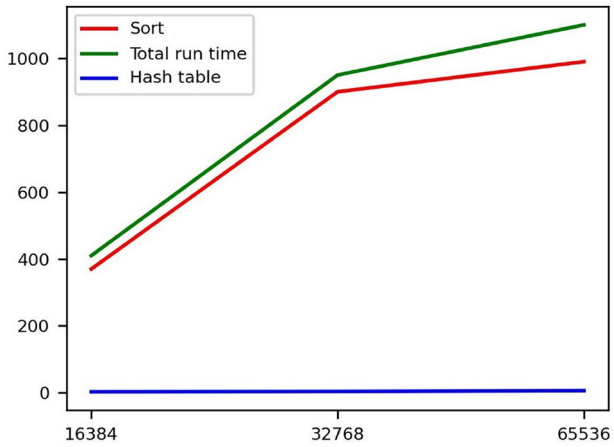


Figure 1. Calculation time for each function in CUDA hash table[7] (*X*-axis : number of particles, *Y*-axis : microseconds (μs)). Microseconds measured with increasing number of particles : Total run time (398, 937, 1057), Sort (367, 906, 1012), Hash table (10.4, 10.7, 11).

framework to optimize sorting and, thereby, improve the performance of GPU-based hashing. Consequently, this has been applied to the cloth-liquid interaction to enhance overall performance.

While recent advances in SPH-based fluid simulation, cloth dynamics, and GPU acceleration have significantly improved individual components of cloth–fluid interaction, their direct combination alone does not resolve the fundamental stability and integration challenges that arise in porous wet-cloth simulation. In particular, existing approaches often model absorption, emission, or diffusion in isolation, rely on unstable pressure estimates near boundaries, or treat GPU acceleration as a collection of local optimizations rather than as a unified pipeline.

In contrast, this work introduces a stability-driven, physically interpretable integration framework that reformulates porous cloth–fluid interaction as a sequential physical process consisting of porous flow, absorption, emission, and diffusion. A key novelty lies in reinterpreting Darcy's law as a directional principle rather than a velocity computation model, enabling robust flow direction estimation without propagating SPH pressure instability. Combined with saturation-centered mass management and a fully GPU-resident interaction pipeline, the proposed framework goes beyond assembling existing techniques and instead establishes a coherent design paradigm for stable, high-performance, and interpretable wet-cloth simulation.

Contributions

(i) Framework-level: we reformulate porous cloth–fluid interaction as a sequential process (porous flow → absorption → emission → diffusion) and introduce a Darcy-inspired, virtual-pressure direction model for stable absorption/emission direction estimation. (ii) Implementation-level: we realize an end-to-end GPU-resident coupling pipeline using Bitonic-sort hashing for scalable neighbor search, and quantify performance and stability through module-wise timing, scaling behavior, and mass-conservation analysis.

Related work

Early research on the effects of liquids includes studies simulating paint flowing on a plane. Initial research on the wet state of liquids includes studies that simulate the flow of paint on a plane. Curtis et al. simulated shallow water in paper fibers for watercolor effects and solved the diffusion equation to handle capillary effects[8]. Subsequently, Chu and Tai proposed a sophisticated system capable of simulating the ink penetration process[9]. Huber et al. added gravity to solve Fick's second law for cloth and simulated the surface absorption of liquid instead of calculating a simple diffusion flow[1]. Although fluid-solid interaction appears in various fields, most studies have simulated individual phenomena rather than integrated results. Ozgen et al. proposed the deformation of cloth submerged in water[10], and Chen et al. suggested improved saturation, wrinkling, and friction models to better represent the shape of wet cloth[11]. Um et al. combined a shallow water model and diffusion equations to depict the dynamic movement of the cloth and the flow of liquid[12].

Lenaerts et al. proposed a method to simulate deformation of cloth due to wetting[13]. They used the SPH approach to represent fluid flow within porous media. Similarly, Rungjiratananon et al. simulated the interaction between fluid and dynamic porous media using SPH and a grid-based data structure[14]. Saket and Parag proposed a geometric diffusion method to efficiently solve the simulation of wet cloth based on the SPH model[15]. Lin proposed a porous model to efficiently represent cohesion and adhesion based on SPH to design a framework to simulate the liquid-hair interaction[16,17]. Fei et al. assumed hair as an impermeable material and proposed a method to detail the interaction with hair using an APIC (Affine Particle-In-Cell)-based liquid simulation[2].

The mixture theory was developed for saturated porous media with incompressible solids, which encompasses the interaction between porous solids and liquids[3]. Abe et al. used the Material Point Method (MPM) to simulate creeping flow in porous soil[18]. Extending this method, Bandara and Soga proposed an algorithm that considers the elastic effects of liquid to simulate porous media capable of large deformations[19]. Daviet and Bertails-Descoubes used an implicit non-smooth approach to simulate the flow of submerged sand grains, employing

mixture theory and the Drucker-Prager model[20]. As mentioned previously, previous techniques aimed to represent individual characteristics of the wet state but did not integrate all the physical features of wet cloth on a GPU. This paper demonstrates that by efficiently modifying each algorithm to be GPU-friendly, the changes and characteristics of materials in the wet state can be implemented and integrated into a single framework.

Xie et al. proposed a novel method for two-way coupling simulation of SPH fluids and FEM solids using a Lagrangian approach[21]. They addressed frictional and non-penetration contacts between SPH fluids and FEM solids through barrier functions and optimization-based time integration. Furthermore, they introduced a time-splitting method and contact proxies to enhance computational efficiency and stability. The use of contact proxies enables simulations that are 2–3 times faster while accurately preserving effects such as non-penetration and friction. This approach also allows for unified treatment of SPH fluids and FEM solids within a single Lagrangian framework. However, in contact-heavy scenarios, the barrier energy sharpness necessitates multiple Newton iterations, leading to increased computational cost.

Li and Desbrun addressed fluid-solid interaction in scenarios with significant density differences, such as water–air systems, using a kinetic solver based on the Lattice Boltzmann Method (LBM). They employed a phase field model to track the interface, allowing surface tension to be computed without the need for complex boundary conditions. In addition, they introduced the Hybrid Moving Bounce-Back (HMBB) technique to ensure stable reflection and momentum transfer at solid boundaries. As a result, by combining traditional grid-based gradients with central difference schemes, the method enhances both accuracy and stability on the normalized interface. However, the dead cell treatment used in this method does not guarantee strict mass conservation, supports only rigid bodies, and does not accommodate coupling with deformable solids[22].

Accurate and efficient simulation of interactions between SPH fluids and deformable solids presents significant computational challenges and limitations in handling complex boundaries compared to conventional approaches. Recently, Ma et al. proposed a method based on Signed Distance Fields (SDFs) and Volume Map that improves efficiency by adopting a local update strategy, rather than globally updating the entire SDF domain as in traditional methods. Their local SDF update detects changes in the positions of triangle patches and selectively updates only the affected regions. Moreover, the Volume Map was approximated using a cosine-based extension function and Monte Carlo integration, instead of a cubic function. As a result, their method achieved up to a 36-fold speedup in SDF and Volume Map updates compared to previous techniques. However, the approach has limitations: storage costs for volume and distance fields increase with higher grid resolutions, and large deformations of the solid may introduce numerical errors[23].

In summary, prior studies on wet cloth and cloth–fluid interaction have achieved meaningful progress in modeling individual components such as absorption, diffusion, porous flow, or GPU acceleration. However, most existing approaches share common limitations, including the lack of integrated treatment of these physical phenomena, numerical instability arising from SPH pressure–based computations, or GPU optimizations that remain partial rather than pipeline-wide. As a result, it has been difficult to simultaneously satisfy stability, interpretability, and performance in high-resolution settings or complex interaction scenarios.

Building upon the strengths of prior work, this study focuses on addressing the structural issues that have remained unresolved. Specifically, we reinterpret Darcy-based directionality to avoid pressure-induced instability, adopt saturation-centered mass management to decouple absorption, emission, and diffusion into sequential stages, and integrate these processes within a fully GPU-resident pipeline. From this perspective, our contribution is not a mere extension of existing functionality, but an integrated design paradigm that fills a critical gap identified in the prior literature.

Summary and comparison

Prior studies on wet cloth and porous media have made substantial progress in modeling individual effects such as absorption, diffusion, and coupling dynamics. However, these approaches differ in how they combine physical processes, handle flow direction, and utilize GPU acceleration. Table 1 provides a qualitative comparison of these structural characteristics.

Specifically, the *Abs./Emi./Diff. Integrated* column indicates whether absorption, emission, and diffusion are modeled within a unified framework or treated separately. The *Stable Flow Dir.* column reflects whether the method explicitly addresses the potential instability of SPH-based pressure near boundaries, as discussed in prior work. The *GPU Pipeline* column describes whether GPU acceleration is applied to isolated components or implemented as an end-to-end pipeline. Finally, the *Interpretable* column refers to whether the formulation provides physically interpretable intermediate quantities, based on how explicitly the underlying processes are modeled.

Method	Abs./Emi./Diff. Integrated	Stable Flow Dir.	GPU Pipeline	Interpretable
Curtis et al. (1997)[8]	No	No	No	Yes
Huber et al. (2011)[1]	Yes	No	No	Yes
Lenaerts et al. (2008)[13]	Yes	No	No	Yes
Fei et al. (2018)[2]	Yes	Yes	No	No
Ours	Yes	Yes	Yes	Yes

Table 1. Comparison with existing cloth–fluid interaction methods. Abs., Emi., and Diff. denote absorption, emission, and diffusion, respectively.

From this perspective, Table 1 is intended to highlight structural differences among existing approaches rather than to provide a strict quantitative ranking. The proposed method integrates porous flow, absorption, emission, and diffusion within a single GPU pipeline and introduces a virtual-pressure-based direction model, representing one possible design choice among these alternatives.

The effectiveness of these design choices is not inferred solely from the comparison in Table 1, but is evaluated quantitatively in later sections using metrics such as sorting cost, frame time, scalability with respect to particle count, and mass conservation error.

Proposed framework

Algorithm overview

The proposed framework models cloth–liquid interaction as a sequential mass-exchange process in porous media, executed at each simulation frame. Such porous-media formulations have been widely adopted to represent fluid transport within fibrous or granular materials[13,24], where absorption, emission, and diffusion are treated as coupled processes.

Starting from a standard SPH fluid update[25], the algorithm first estimates a stable internal flow direction using a virtual-pressure formulation inspired by Darcy's law, which has been used in prior work to model flow behavior in porous media while avoiding instability of pressure-based formulations[13,24]. Based on this direction, fluid mass is transferred into the porous cloth through a saturation-aware absorption step, which reflects the storage capacity of the material and is consistent with mixture-based porous modeling approaches[1]. When local saturation exceeds a prescribed threshold, excess mass is released via an emission step to maintain bounded mass behavior. This is followed by a diffusion stage that redistributes absorbed mass among neighboring porous elements to ensure spatial smoothness, as commonly adopted in particle-based diffusion models[1,12].

Each stage operates on local neighborhoods and is executed explicitly, forming a clear pipeline of *porous flow* → *absorption* → *emission* → *diffusion*, which underlies the equations introduced in the following subsections.

We used the method proposed by Akinci et al.[26] for particle sampling of cloth and rigid bodies, and the interaction with fluid particles. The particles sampled in the cloth are referred to as porous particles. Before explaining the details, the notation is described. The nodes of the cloth are denoted by n, fluid particles by f, and the porous particles by p. V_i represents the volume of the particle, calculated using the method proposed by Akinci et al. for boundary particles, and $\frac{m_i}{\rho_i}$ for fluid particles. Φ is the porosity of the solid within the porous material and s_i is the saturation fraction representing the amount of absorbed fluid within the porous material. The remaining notations are as shown in Table 2.

Figure 2 shows the algorithm overview of the method proposed in this paper. The preprocessing step involves transferring the face array and boundary particles required for cloth simulation, as well as the particle array required for liquid simulation, from the CPU to the GPU. On the GPU, a hash table necessary for rapid dynamics calculations is constructed based on Bitonic sort. In Bitonic sort, the unsorted list (cell id, particle id) is sorted based on the cell id, allowing quick access to neighbor particles within the CUDA kernel.

In the main loop calculated for every frame, the dynamics of the cloth is calculated based on Projective Dynamics[27], and the dynamics of the fluids is calculated on CUDA based on DFSPH (Divergence-free SPH)[25]. During this process, interaction (i.e., two-way coupling) requires access to neighboring particles, and this is efficiently computed on the GPU using a hash table updated with CUDA-based Bitonic sort. Based on this data structure, absorption, emission, and diffusion expressed in the cloth-liquid interaction are stably simulated on CUDA. Finally, the computed GPU memory is transferred to the CPU for visualization with hardware-accelerated shading. This paper proposes an integrated framework on the GPU for various effects expressed through the interaction of fluid and cloth.

Symbol	Location	Description
h_i	Particle	Support radius
r_i	Particle	Radius
x_i	–	Position
v_i	–	Velocity
$v_{i,f}$	Porous particle	Pore flow
V_i	Particle	Volume
m_i	–	Total mass
$m_{i,f}$	Node, Porous particle	Fluid mass
$m_{i,s}$	Node, Porous particle	Solid mass
w_i	Porous particle	Barycentric coordinate
ρ_i	SPH, Porous particle	Density
ρ_i^0	–	Rest density
Φ_i	Node, Porous particle	Solid fraction
s_i	Node, Porous particle	Saturation

Table 2. List of symbols used in the paper.

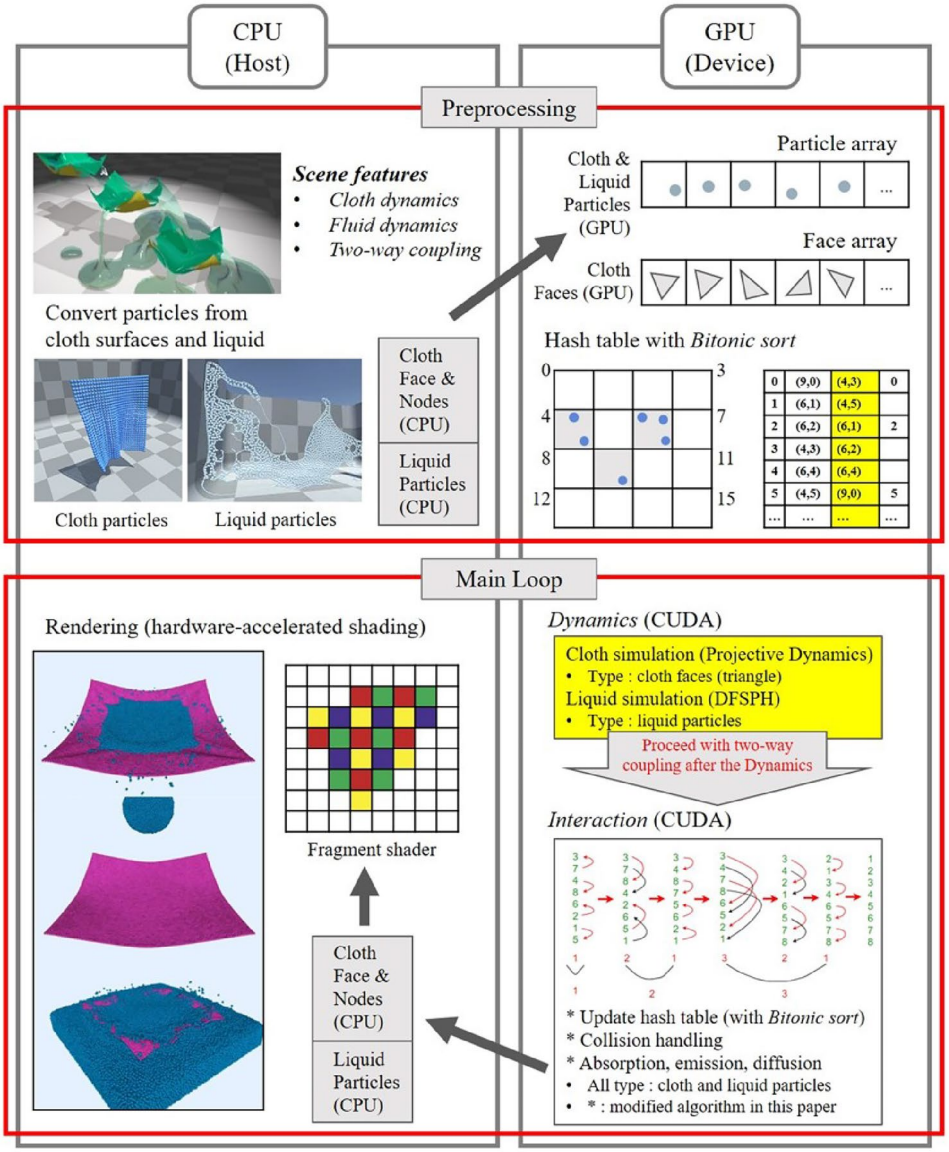


Figure 2. Algorithm overview of proposed method.

Unified pipeline overview. For clarity, we provide a unified step-by-step overview of the proposed framework. At each simulation frame, the algorithm proceeds as follows:

1. Neighbor search is performed using a GPU-based hash table constructed via Bitonic sort.
2. Cloth dynamics are updated using Projective Dynamics, and fluid dynamics are computed using DFSPH.
3. A stable flow direction is estimated for each porous particle using the virtual-pressure-based formulation.
4. Fluid mass is transferred from fluid particles to porous particles through the saturation-driven absorption process.
5. If local saturation exceeds the threshold, excess mass is emitted back to fluid particles.
6. The absorbed mass is redistributed among neighboring porous particles via diffusion to ensure spatial smoothness.
7. The updated mass and saturation values are propagated to cloth nodes for subsequent simulation steps.

This unified description complements the detailed explanations provided in the following subsections and clarifies the overall data flow and execution order of the algorithm.

Porosity modeling

First, the mass of the fluid absorbed by the cloth is stored in the nodes of the cloth mesh. The maximum mass of fluid that can be absorbed by each node is then calculated through the porous particles sampled in the cloth. This is used to calculate the saturation of the nodes (see Fig. 3a).

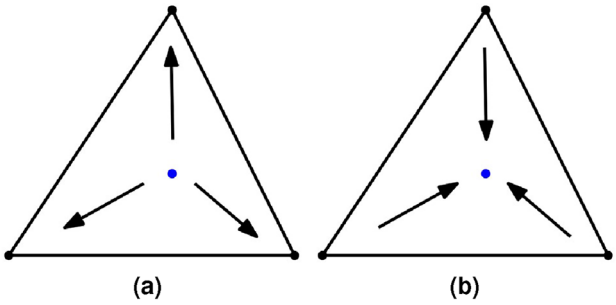


Figure 3. Property transfer between porous particles and nodes in a triangle.

The saturation of a node is calculated from the porous particle *i* sampled on the neighboring face as follows (see Eqs. 1 and 2).

$$m_{n,fmax} = \sum_i w_i \left(1 - \Phi_i\right) V_i \rho_f^0 \tag{1}$$

$$s_n = \frac{m_{n,f}}{m_{n,fmax}} \tag{2}$$

In the above equations, $m_{n,fmax}$ represents the maximum mass of liquid that can be absorbed by each node. $(1 - \Phi_i)\, V_i$ represents the volume of fluid within the porous particle, and multiplying this by ρ_f^0 gives the maximum mass of liquid that can be absorbed by the porous particle. By multiplying this by the weight coordinate w_i of the porous particle and adding it for each node, we can calculate the maximum mass of liquid that can be absorbed by each node. Dividing the current mass of the liquid held by the node by this value gives the saturation of the node.

The calculated saturation of the node is interpolated to the surrounding sampled porous particles to determine the saturation of the porous particles (see Fig. 3b). Equation 3 is used to calculate the saturation of the porous particles.

$$s_p = s_0 w_0 + s_1 w_1 + s_2 w_2 \tag{3}$$

where 0∼2 represents the three nodes of the face where the particles are sampled, and *w* denotes the barycentric coordinates of the face.

Pore flow

The direction of fluid absorption and emission is modeled based on a formulation inspired by Darcy's law, which is commonly used to describe flow behavior in porous media[13,24]. In standard formulations, Darcy's law computes flow velocity from pressure gradients and external forces. However, in SPH-based simulations, pressure estimation near boundaries is known to be unstable due to irregular particle distributions[13,25]. To address this issue, we do not directly use the pressure magnitude for velocity computation. Instead, we adopt a virtual-pressure-based formulation to estimate only the flow direction, which improves numerical stability while preserving the physical intuition of Darcy-driven flow. Bender et al.[25] proposed a formulation for computing fluid velocity in porous materials based on gravity and pressure, as shown in Eq. (4).

$$v_p = -\frac{K_i}{\phi_i \mu} \left(\nabla P - \rho g\right) \tag{4}$$

However, the pressure of fluid particles calculated using the SPH method is unstable. Therefore, in this study, we improve stability by calculating only the direction of fluid flow within the porous material using virtual pressure instead of the actual pressure of the particles. The virtual pressure is calculated as follows (see Eq. 5).

$$P' = k^p \sum_i V_i \nabla W_{p,i} \tag{5}$$

where P' denotes the virtual pressure, k^p is a user-specified coefficient, V_i is the volume of boundary particle *i*, and $\nabla W_{p,i}$ is the gradient of the SPH kernel evaluated between porous particle *p* and boundary particle *i*. Here, *i* indexes the boundary particles, excluding both porous particles and fluid particles neighboring the porous particles. Using this pressure, the flow direction of the fluid is calculated as follows (see Eq. 6).

$$v_{p,f} = \frac{P' + g}{\|P' + g\| + \epsilon} \tag{6}$$

where $v_{p,f}$ denotes the normalized flow direction of the fluid at porous particle *p*, P' is the virtual pressure defined in Eq. (5), *g* is the gravity vector, and ϵ is a small constant introduced to avoid division by zero.

The reason for adding ϵ to the denominator is to ensure that $v_{p,f}$ becomes a zero vector when the value of $|P' + g|$ approaches zero. When $v_{p,f}$ is a zero vector, that is, there is no fluid flow, it indicates that there is no interaction between the porous particle and the fluid particle. The fluid flow calculated in this way affects the absorption of water.

Absorption

In this study, we model the mass transfer from fluid particles to porous particles using an SPH-based formulation (see Fig. 4), which is commonly adopted in particle-based fluid–solid interaction and porous media simulation[13,24]. This approach enables localized and continuous mass exchange based on particle interactions, which is suitable for representing absorption behavior in discretized porous materials. In particular, SPH provides a natural framework for modeling particle-level interactions and mass exchange in a Lagrangian setting.

When a fluid particle comes into contact with a porous particle, the porous particle absorbs the mass of the fluid particle. The mass absorbed by the porous particle *j* from the neighboring fluid particle *i* is calculated as Eq. 7.

$$\Delta m_i = \sum_j k_j^a \left(s_{j,\max} - s_j + (s_{j,\max} - 1)\cos\theta_{ij}\right) V_j \nabla^2 W_{i,j} \tag{7}$$

where Δm_i denotes the amount of mass absorbed from fluid particle *i*, k_j^a is an absorption coefficient controlling the rate of mass transfer, s_j is the saturation of porous particle *j*, and $s_{j,\max}$ is its maximum saturation. The term $\cos\theta_{ij}$ represents the directional alignment between the vector $(x_i - x_j)$ and the internal flow direction of porous particle *j*, where θ_{ij} is the angle between them. V_j is the volume of porous particle *j*, and $\nabla^2 W_{i,j}$ denotes the Laplacian of the SPH kernel evaluated between particles *i* and *j*.

Additionally, to adjust the degree of absorption by the porous particle according to the fluid flow direction, a new element $(s_{j,max} - 1)\cos\theta_{ij}$ is added. $(s_{j,max} - 1)\cos\theta_{ij}$ signifies the influence on fluid flow within the porous particle. Here, θ_{ij} is the angle between $x_i - x_j$ and $v_{j,f}$. fluid particles located in the opposite direction of the fluid flow cannot pass through because the porous particle can only absorb up to a saturation of 1. Conversely, fluid particles in the direction of the fluid flow can absorb an additional $1 - s_{j,max}$. Finally, the mass of fluid absorbed by the porous particle is interpolated to the nodes and added to the fluid mass of the node. Equation 7 is not derived from a strict continuum formulation, but rather constructed as a physically motivated approximation that combines saturation effects, directional alignment, and SPH-based spatial weighting.

Parameter rationale. The coefficients k^a and k^d are introduced as *dimensionless rate parameters* that control the temporal and spatial smoothing of absorption and diffusion, respectively, rather than as material-specific physical constants. Specifically, k^a regulates how rapidly saturation-driven absorption is applied per time step, while k^d controls the smoothness of mass redistribution during diffusion. These parameters do not alter mass conservation or flow directionality, and are chosen within ranges that ensure numerical stability and bounded saturation values. In practice, the framework exhibits low sensitivity to moderate variations of k^a and k^d, as changes primarily affect the rate of convergence rather than the qualitative behavior or final equilibrium state.

If the change in mass Δm_i is greater than or equal to $m_i - m_{f,min}$, then m_i is used instead of Δm_i . If subtracting Δm_i from m_i results in m_i becoming 0, the fluid particle is removed from the simulation.

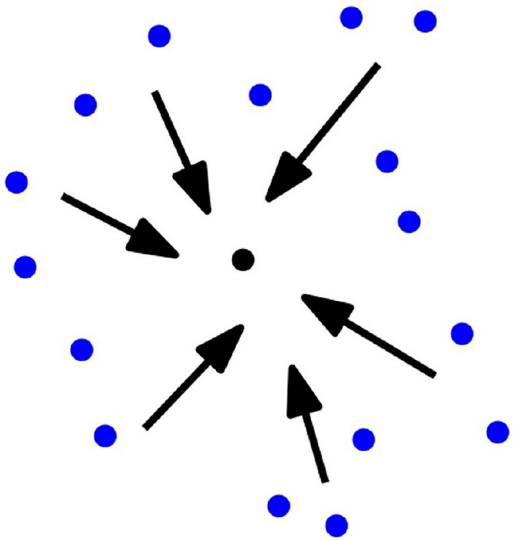


Figure 4. Calculation of the absorbed mass from fluid particles to porous particles (blue: fluid particle, black: porous particle).

Otherwise, the h_i and r_i of the fluid particle are adjusted in proportion to $\frac{m_i}{m_{f,max}}$. $m_{f,min}$ and $m_{f,max}$ are the minimum and maximum masses of the fluid particle. The calculated Δm_i is transferred to *j*, and divided according to the contribution ratio to Δm_i. The Δm_i transferred to the porous particle is then passed to the nodes of the cloth as Eq 8. Consequently, this process interpolates the mass of the fluid absorbed by the porous particles and adds it to the fluid mass of the nodes.

$$\Delta m_n = \sum_j w_j \Delta m_j \tag{8}$$

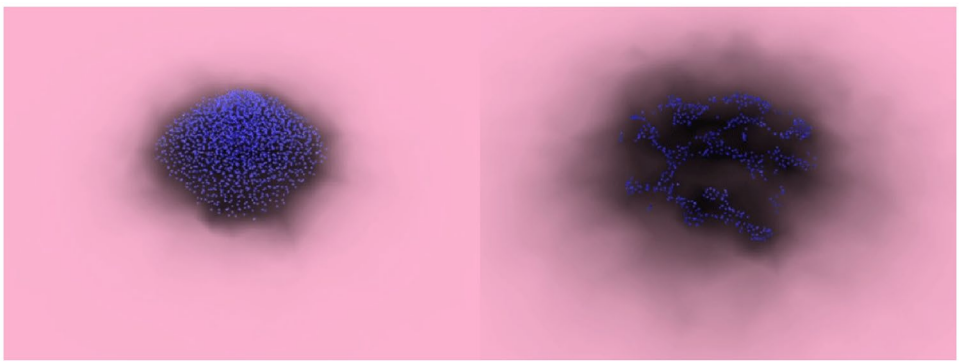
where Δm_n denotes the change in fluid mass at node *n*, w_j is the interpolation weight associated with porous particle *j*, and Δm_j is the mass change of porous particle *j*. Here, *j* indexes the porous particles sampled around node *n*. The updated node mass is given by $m_{n,f} \leftarrow m_{n,f} + \Delta m_n$, and the saturation at node *n* is adjusted accordingly.

In this paper, the "previous method" refers to a baseline SPH-based cloth–fluid interaction model inspired by Fei et al.[2], which does not incorporate virtual-pressure-based flow direction, saturation-driven mass control, or directional absorption and emission terms.

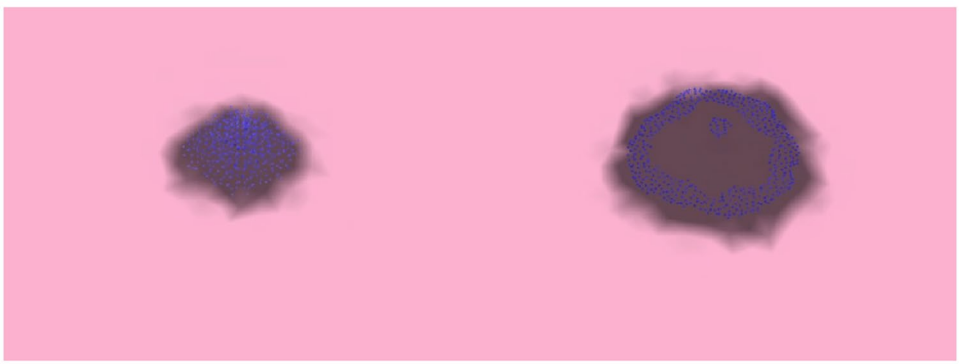
The comparison with Fei et al. is performed under identical simulation settings within the same implementation framework, varying only the inclusion of the proposed components. Therefore, the results should be interpreted as a relative comparison under controlled conditions rather than an absolute performance benchmark against the original implementation.

Figure 5 shows the comparison of absorption effects between the previous method and our method. In this experiment, only the absorption resulting from the cloth-liquid interaction was calculated, and diffusion has not yet been included. In the previous method, the absorption process resulted in somewhat unnatural outcomes due to the clumping of water particles (see Fig. 5a). However, in our method, the movement of the water particles absorbed during the cloth-liquid interaction is naturally represented (see Fig. 5b).

To facilitate a more objective comparison, Table 3 summarizes the simulation parameters and key outcome indicators corresponding to Fig. 5. The table confirms that both methods were evaluated under identical conditions, while highlighting the improved stability and reduced particle clumping achieved by the proposed absorption model.



(a) Previous method[2]



(b) Our method

Figure 5. Comparison of absorption behavior in cloth–liquid interaction (pink: cloth surface, particles: water). Both methods are evaluated under identical conditions (time step $\Delta t = 0.004$ s; $N_f = 200{,}800$ fluid particles; $N_c = 9{,}022$ cloth nodes; same scene initialization). Quantitative settings and outcome indicators are summarized in Table 3.

Here, $s_{max} > 1.0$ does not represent a physically exact saturation limit, but rather a relaxed numerical bound introduced to ensure stable and continuous mass transfer in the discretized SPH framework.

Justification of Eqs. 6–8. Equations 6–8 are not intended as strict first-principles derivations from continuum mechanics, but rather as *physically motivated modeling approximations* designed for stable cloth–fluid interaction in an SPH setting. Darcy's law provides the physical intuition that flow in porous media follows the direction of decreasing pressure and gravity; however, directly using SPH pressure magnitudes near boundaries is known to cause numerical instability. Accordingly, Eq. 6 preserves only the *directional component* of Darcy's law by defining a virtual pressure gradient based on kernel distributions and normalizing it to obtain a stable flow direction, an approach consistent with prior porous-flow and hair–fluid interaction models.

Equation 7 combines three physically meaningful factors: saturation deficit, directional alignment between fluid motion and internal porous flow, and SPH kernel-based spatial weighting. This formulation ensures that absorption is bounded by storage capacity, biased along physically plausible flow directions, and localized according to particle proximity, while maintaining mass conservation. Finally, Eq. 8 discretely transfers absorbed mass from porous particles to cloth nodes via weighted accumulation, representing a consistent discrete realization of mass conservation commonly adopted in SPH–solid coupling frameworks. Together, Eqs. 6–8 form a stable and interpretable approximation that preserves the essential physics of porous flow while remaining robust in GPU-based SPH simulations.

Emission

If the fluid mass at a node exceeds the maximum allowable mass, the excess amount is redistributed to the surrounding porous particles through interpolation. This process is designed to maintain mass conservation and ensure stable particle-based mass exchange, which is commonly considered in SPH-based fluid–solid interaction models[13,25].

To preserve mass consistency, the redistribution is guided by the center of mass computed from neighboring particles. If fluid particles exist along the emission direction of a porous particle, the excess mass is transferred to those particles while ensuring that their mass remains within a prescribed upper bound. If the remaining excess mass exceeds the minimum allowable mass of a fluid particle, a new particle is generated along the emission direction.

Furthermore, emission is restricted in the presence of boundary particles to avoid non-physical penetration. If the boundary particle corresponds to a porous particle, the excess mass is instead transferred to that particle. Finally, the redistributed mass is interpolated back to the node and reflected in its fluid mass, ensuring consistency in the overall mass distribution.

The detailed process is as follows. If the saturation of a node exceeds 1, fluid particles are emitted. The fluid mass $\Delta m_n = m_{n,f} - m_{n,fmax}$ to be emitted from the node is transferred to the sampled porous particle *i* as Eqs 9 and 10.

$$w'_n = \sum_j w_j \tag{9}$$

$$\Delta m_i = \sum_{k=0}^{2} w_k \frac{\Delta m_k}{w'_k} \tag{10}$$

where w'_n denotes the normalization factor computed as the sum of interpolation weights w_j for the porous particles surrounding node *n*, and w_j is the interpolation weight associated with porous particle *j*. In Eq. (10), Δm_i represents the redistributed mass, and the index $k \in \{0, 1, 2\}$ refers to the three porous particles associated with the triangle surrounding the node. Here, w_k is the interpolation weight and Δm_k is the mass change of porous particle *k*, while w'_k is the corresponding normalization factor ensuring proper weight normalization.

In Eq 9, *j* is the porous particle sampled on the neighboring face of the node. After calculating Δm_i, if the angle between the vector connecting the porous particle *i* and the neighboring fluid particle *j* and $v_{i,f}$ is less than θ^e, and m_j is less than m_{fmax}, a mass change amount less than $m_{fmax} - m_f$ is transferred to *j*. Furthermore, if Δm_i is greater than m_{fmin}, fluid particles of the remaining mass are emitted in the direction of $v_{i,f}$.

	Previous method	Proposed method
Number of fluid particles	200,800	200,800
Number of cloth nodes	9,022	9,022
Time step	0.004 s	0.004 s
Absorption coefficient k_a	Fixed	0.01
Max. saturation $s_{\max}$	1.0	1.18
Directional absorption	No	Yes
Particle clumping	High	Low
Mass conservation	Partial	Preserved
Absorption stability	Unstable	Stable

Table 3. Simulation parameters and quantitative indicators for Fig. 5 (Absorption comparison).

If the angle between the direction vector to the neighboring boundary particle j and the emission direction is less than θ^e, no emission occurs. If the boundary particle is a porous particle, the remaining fluid change is transferred as the mass change of that particle. Finally, the mass change is transferred to the node as described in Section 'Absorption'. During this process, the fluid particles can become very dense when they are emitted. To address this issue, the simulation is stabilized by limiting the density and velocity for a certain number of frames, as proposed in a previous method[28].

Figure 6 shows the comparison of emission effects using the previous method and our method. In the case of the previous method, the unnatural movement of the water particles observed in absorption also affects emission, resulting in unnatural outcomes (see Fig. 6a). This can affect the physical quantities transferred to the cloth surfaces, sometimes resulting in blotchy patterns. In contrast, our method effectively represents the natural emission of liquid without clumping particles and, even without the diffusion process, demonstrates isotropic transfer of physical quantities to cloth surfaces without blotchy patterns (see Fig. 6b).

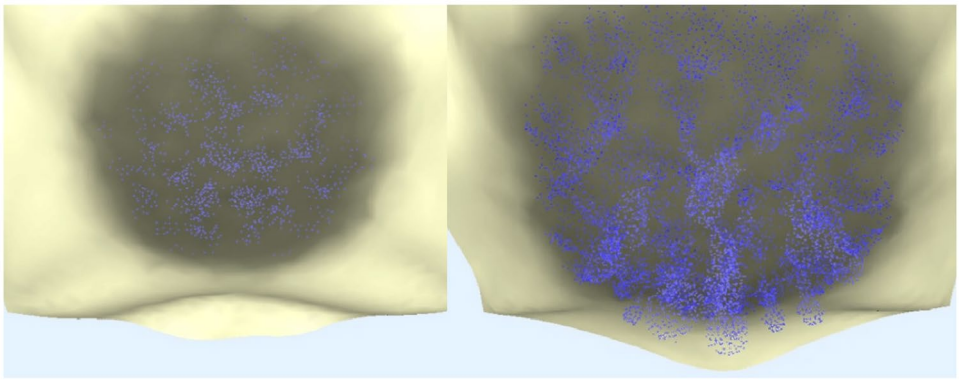
For clarity, Table 4 reports the simulation parameters and qualitative performance indicators associated with Fig. 6. This tabulated comparison enables a clearer assessment of emission behavior, showing that the proposed method achieves more isotropic and mass-preserving emission under the same experimental settings.

Diffusion

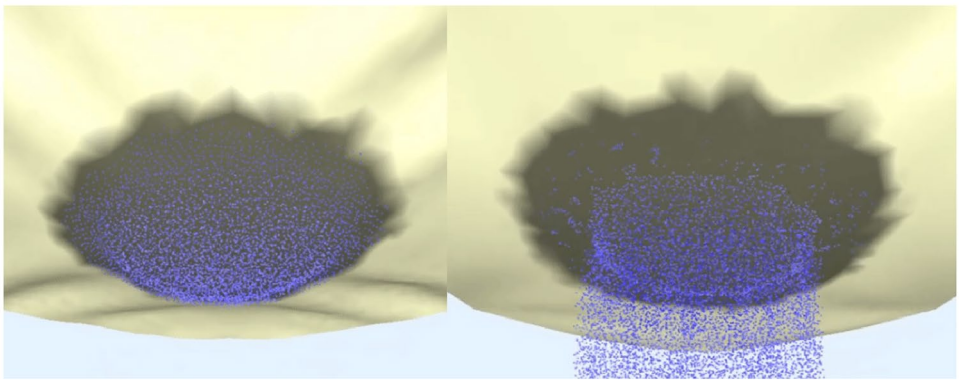
Finally, the diffusion process is applied after absorption and emission to redistribute the absorbed fluid within the porous material. This sequential treatment allows the model to first handle mass transfer events and then enforce spatial smoothness, which is commonly adopted in particle-based diffusion modeling[1,13].

After computing absorption and emission (Eq. 3), the saturation of each porous particle is updated. The fluid mass stored at the nodes, denoted by $m_{n,f}$, is then transferred to the corresponding porous particles using interpolation. Based on these updated values, diffusion is computed using an SPH-based formulation, where mass exchange occurs between neighboring porous particles according to local saturation differences and spatial proximity.

This formulation enables continuous and localized redistribution of mass, ensuring smooth wetting patterns while maintaining consistency with the particle-based representation of the porous medium.



(a) Previous method[2]



(b) Our method

Figure 6. Comparison of emission behavior in cloth–liquid interaction (light yellow: cloth surface, particles: water). Both methods are evaluated under identical conditions (time step $\Delta t = 0.004$ s; $N_f = 610{,}048$ fluid particles; $N_c = 13{,}978$ cloth nodes; same scene initialization). Quantitative settings and outcome indicators are summarized in Table 4.

	Previous method	Proposed method
Number of fluid particles	610,048	610,048
Number of cloth nodes	13,978	13,978
Time step	0.004 s	0.004 s
Emission threshold	Heuristic	Saturation-based
Flow-direction check	No	Yes
Particle clustering after emission	Severe	Minimal
Isotropic emission	No	Yes
Mass preservation	Partial	Preserved

Table 4. Simulation parameters and quantitative indicators for Fig. 6 (emission comparison).

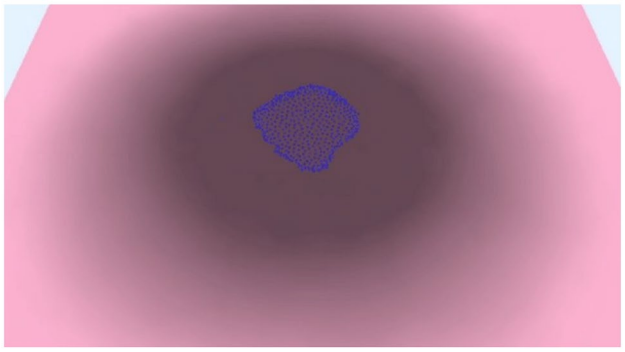


Figure 7. Diffusion effects in the proposed cloth–liquid interaction framework (pink: cloth surface, particles: water). The result is generated with the same simulation settings (time step $\Delta t = 0.004$ s and the parameters reported therein), where diffusion is applied after absorption and emission to produce spatially smooth wetting patterns.

The mass change of the porous particle *i* due to diffusion from the neighboring porous particle *j* is calculated as Eq. 11.

$$\Delta m_i = -\Delta t\, m_{i,f}\, k_j^d \left(s_i - s_j + \eta^d \cos\theta_{ij}\right) V_j \nabla^2 W_{i,j} \tag{11}$$

where Δm_i denotes the mass change of porous particle *i* due to diffusion, Δt is the simulation time step, and $m_{i,f}$ is the fluid mass associated with porous particle *i*. The coefficient k_j^d controls the diffusion rate, and η^d is a parameter that determines the influence of gravity on the diffusion process.

Here, s_i and s_j denote the saturation values of porous particles *i* and *j*, respectively, and θ_{ij} is the angle between the vector $(x_i - x_j)$ and the direction of gravity. V_j is the volume of porous particle *j*, and $\nabla^2 W_{i,j}$ denotes the Laplacian of the SPH kernel evaluated between particles *i* and *j*.

The calculated mass change is interpolated to the node in the same manner as before and added to the fluid mass of the node. This diffusion calculation can become unstable if the timestep is too large or the mass change is too great. In this paper, a sub-timestep approach is implemented to efficiently handle such issues.

Figure 7 shows the diffusion effects experimented with our method. Some of the previous experiments did not include diffusion, which resulted in somewhat rough wetting effects. When diffusion is applied as the final stage of the proposed method, the saturation spreads isotropically, resulting in smoother effects.

Optimization of GPU Hashing

Kernel design for GPU-based bitonic sort

Bitonic sort is an algorithm that efficiently performs sorting in a parallel computing environment. In a single-threaded environment, Bitonic sort is less efficient with a performance of $O\left(Nlog^2N\right)$ compared to the typical sorting algorithm performance of $O\left(NlogN\right)$. However, in parallel computing environments with many processors, such as a GPU, it can sort with a performance of $O\left(log^2N\right)$. Bitonic refers to two monotonic sequences in which the sequence increases before a specific element and decreases after it. The key to this algorithm is to divide the data set into two separate sorted parts and then merge these parts to obtain a sorted entire data set. Bitonic sort merges elements organized into Bitonic sequences, maintaining the Bitonic sequence form as a result. This characteristic makes it suitable for parallel programming because performing sorting on either of the two monotonic sequences first does not affect the final result while maintaining the sequence form.

Figure 8 shows the Bitonic sorting process, and its principle is similar to that of the merge sort. However, unlike merge sort, which requires a single thread to check all elements, preventing performance improvement

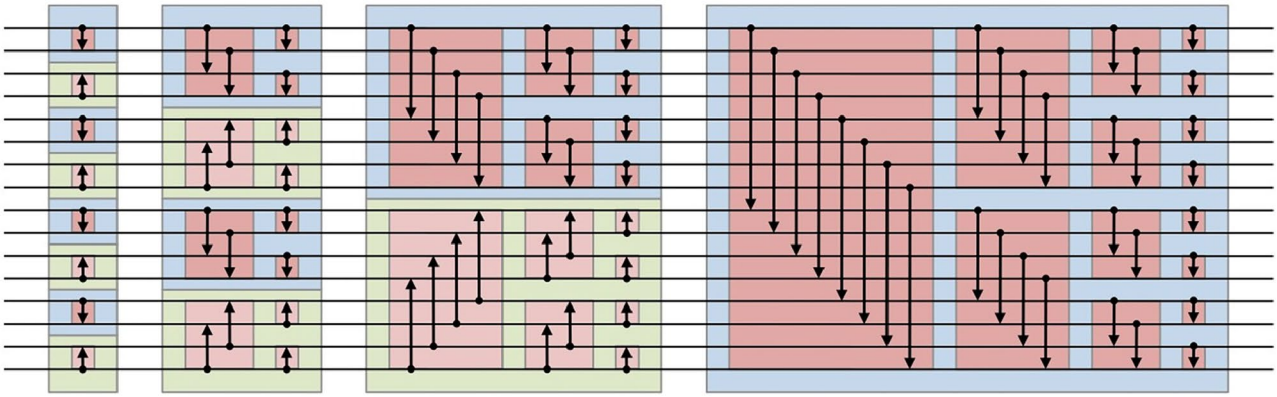


Figure 8. The process of Bitonic sort. Adapted from the public domain image on the Wikipedia article *Bitonicsorter* (https://en.wikipedia.org/wiki/Bitonic_sorter).

h : hash table
N_b : number of GPU blocks
N_t : number of GPU threads
N_f : number of water particles
1: **procedure:** *BitonicSortFunc*()
2: For ($i \leftarrow 2, i \leq N_f, i \leftarrow 2i$)
3: For ($j \leftarrow i/2, i \geq 1, j \leftarrow j/2$)
4: bitonicSortKernel<<<N_b, N_t>>>(h, i, j, N_f);

Table 5. Pseudocode for the Bitonic sort function.

1: **procedure:** *bitonicSortKernel*(h, i, j, N_f)
2: *tid* ←threadIdx.x+blockIdx.x+blockDim.x
3: If *tid* < N_f && *tid*%(j≪1)<j
4: *descending*← (*tid*/i) %2
5: If *descending* && *h.hashing*[tid] < *h.hashing*[tid+j]
6: swap(*h.hashing*[tid], *h.hashing*[tid+j])
7: swap(*h.particleId*[tid], *h.particleId*[tid+j])
8: ElIf !*descending* && *h.hashing*[tid] > *h.hashing*[tid+j]
9: swap(*h.hashing*[tid], *h.hashing*[tid+j])
10: swap(*h.particleId*[tid], *h.particleId*[tid+j])

Table 6. Pseudocode for the GPU kernel function in Bitonic sort.

below $O(N)$ even with many processors, Bitonic sort allows all operations within the vertically aligned red boxes in the figure to be processed in parallel. Therefore, with many processors, the sorting can be achieved in $O\left(log^2 N\right)$ time, which is the sum of 1 to *logN*. For parallel processing, the execution of the Bitonic sort function, the Bitonic merge function, and the loops for comparison must all be executed in parallel across multiple threads.

Table 5 shows the pseudocode for the Bitonic sort function. The nested loop sets the interval of element division i and the element comparison offset j, executing the Bitonic sort function in groups of blue and green boxes (see Fig. 8). In this function, *bitonicSortKernel* is the GPU kernel function, and the detailed GPU kernel is given as Table 6.

In the above code, h is an object containing the hashing data structure. Figure 8 shows the process of the Bitonic merge function, where the hash index and particle index are alternately sorted according to the monotonic sequence corresponding to the thread index, as seen in the blue and green boxes.

Configuring GPU-based hash table

In this section, we explain the method of building a GPU-based hash table using the sort kernel mentioned above. As shown in Fig. 1, the sorting process consumes the most computation during hash table construction, so we replace this part with Bitonic sort, while using NVIDIA's CUDA hash table data structure for the rest[7]. The steps are as follows: 1) Store the particle data in the hash table, 2) sort the data using GPU-based Bitonic sort, and finally 3) reorder the data through the grid index.

Figure 9 compares the time taken for sorting using the GPU-based Thrust library and Bitonic sort. Despite both methods being executed on a GPU, the Thrust method shows an increase in computation time as the number of particles increases, whereas Bitonic sort exhibits relatively little change in computation time.

Performance comparison analyzed using particle-based density calculation

In this section, we demonstrate the efficiency of the data structure described above by applying it to a representative application that uses a hash table: particle-based fluids. We compared the time taken to calculate the density, which utilizes the distribution of neighboring particles, while updating the hash table for every frame as the positions of the particles change. The method for calculating density in particle-based fluids is given by Eq. 12.

$$\rho(x) = \sum_i m_i W(x_i - x, h) \tag{12}$$

where ρ is the density, m is the mass, and W is the W_{poly} smoothing kernel (see Eq. 13).

$$W_{poly}(r,h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - r^2\right)^3 & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{13}$$

where h is the smoothing radius, and r is the distance between the target particle and the neighboring particle.

Figure 10 compares the time taken to calculate density using particle data and a hash table. To produce the same results, our implemented Bitonic sort-based hashing algorithm showed faster performance. Figure 11 compares the density measured in the same frame. The overall trends are consistent between the Thrust library and the Bitonic sort-based hash table, although minor deviations are observed. These differences arise from variations in sorting strategies and numerical precision, which are inherent to parallel SPH implementations. In this study, we optimized the cloth–liquid interaction on a GPU based on this technique.

In this paper, we applied GPU-based Bitonic sort to particle-based water simulations, so it can be utilized in most fields where NNP (Nearest Neighbor Particle) is needed, even beyond fluids. In this paper, we also applied the SPH (Smoothed Particle Hydrodynamics) method to 1) surface reconstruction and 2) turbulent flows, and compared and analyzed the performance.

Figure 12 shows the results of reconstructing fluid surfaces through a particle-based level-set by calculating NNP using GPU-based Bitonic sort[29]. This is visualized as a half-plane in a 3D SPH simulation, with approximately 50,000 particles used in the simulation. The only difference compared to the sort using Thrust is the NNP calculation, and the fluid surface generated from it produced natural results without any visible loss

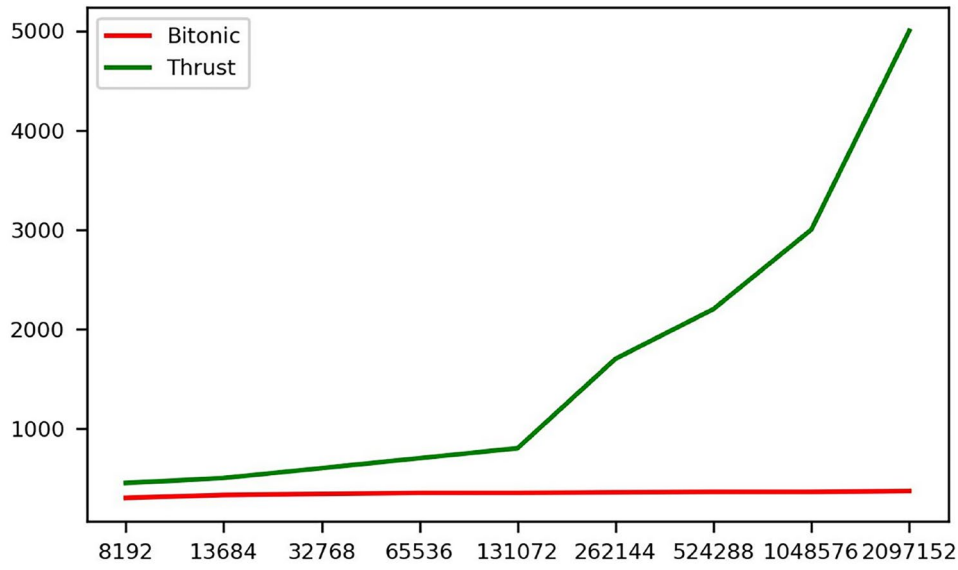


Figure 9. Sorting time comparison for GPU hashing: Thrust-based sorting vs. the proposed Bitonic sort (*X*-axis: number of particles; *Y*-axis: time in μs). All measurements were conducted on the same GPU hardware (GeForce GTX 1080Ti) under identical conditions; only the sorting implementation is changed.

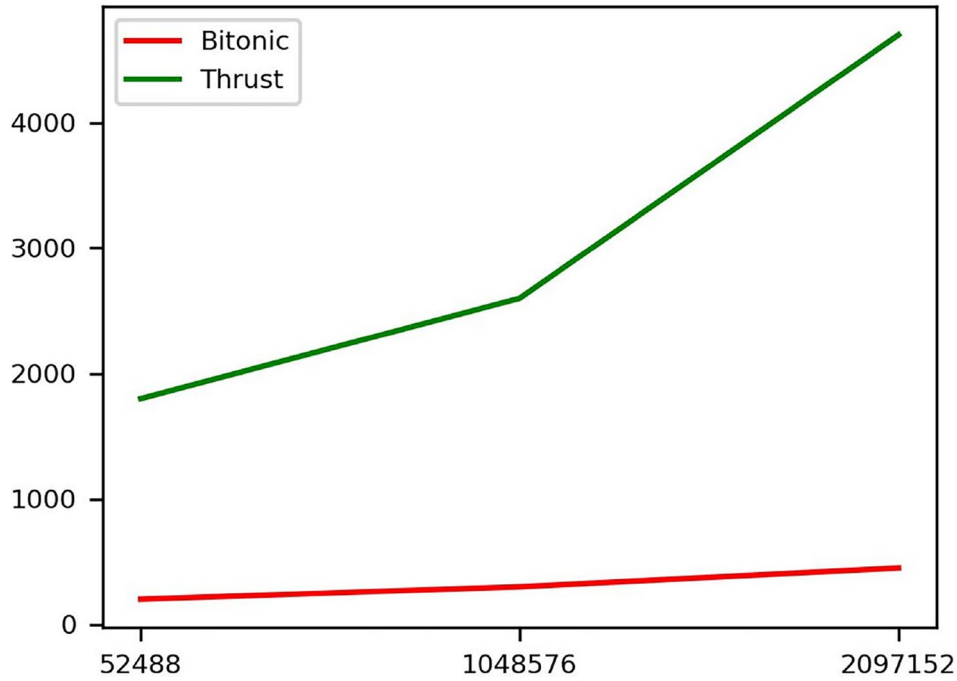
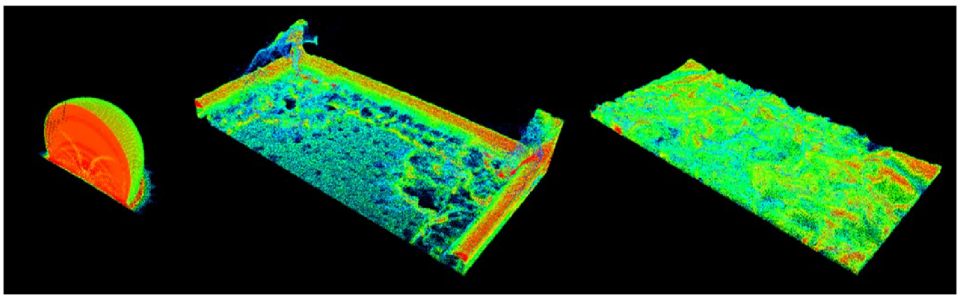
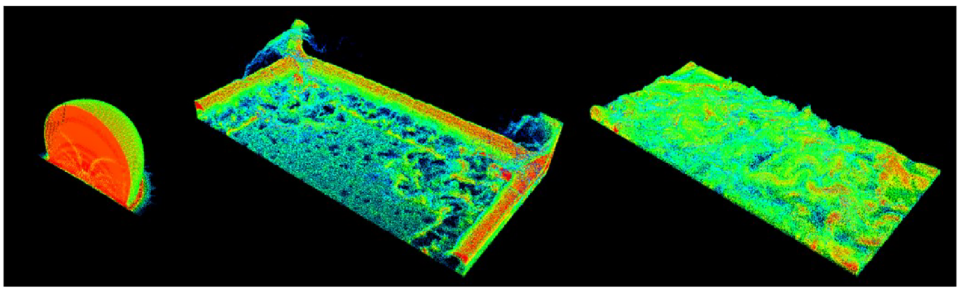


Figure 10. Density computation time comparison using GPU hashing: Thrust-based vs. Bitonic sort–based neighbor search (*X*-axis: number of particles; *Y*-axis: time in μs). All measurements were conducted under identical simulation settings on the same GPU (GeForce GTX 1080Ti); only the neighbor-search/sorting backend differs.



(a) Visualizing density with Bitonic sort based hash table



(b) Visualizing density with Thrust library based hash table[7]

Figure 11. Density visualization comparison for moving particles using (**a**) Bitonic sort–based hashing and (**b**) Thrust-based hashing. Both images are rendered at the same frame with identical particle states and visualization settings. The overall density patterns are consistent, although minor deviations are observed due to differences in sorting strategies and numerical precision inherent to parallel SPH implementations.

Figure 12. SPH surface reconstruction using the proposed Bitonic sort–based hashing for nearest-neighbor particle (NNP) queries. This example uses approximately 50,000 particles under the same simulation and rendering settings as the baseline, and demonstrates that acceleration of neighbor search does not degrade reconstruction quality.

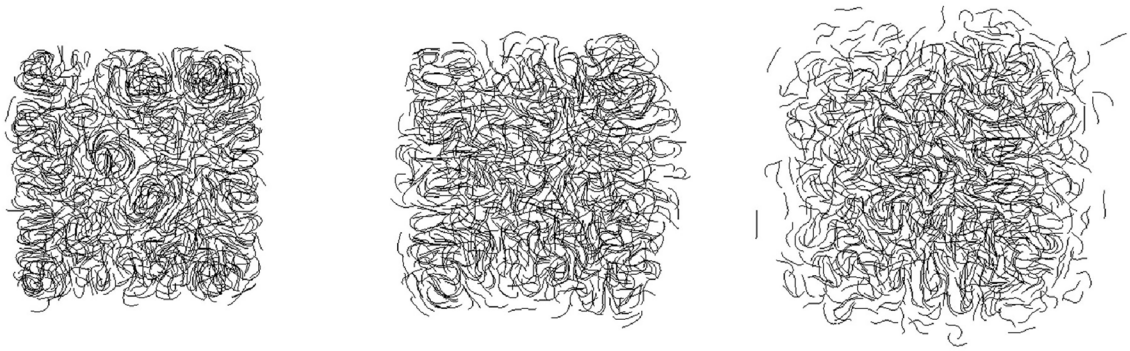


Figure 13. SPH-based turbulent flow simulation using the proposed Bitonic sort–based hashing for NNP queries. The simulation uses identical physical parameters and rendering settings as the baseline, while replacing only the hashing/sorting backend, enabling higher particle counts and improved performance without sacrificing visual fidelity.

when viewed visually. GPU-based Bitonic sort demonstrated approximately 10 times faster performance, which showed similar results to the validation test (density) conducted in Fig. 11.

Figure 13 shows the results of calculating turbulent flow using a particle-based approach[30], and since NNP is also required in this process, we utilized GPU-based Bitonic sort. The turbulent flow is well represented, and compared to the turbulent flow generated using Thrust for sorting, it demonstrated approximately 9 times faster performance. One of the most critical factors in particle-based turbulence representation is the accuracy and efficiency of nearest neighbor particle (NNP) search. In this study, we employed a GPU-based Bitonic sort to construct a hash table, enabling precise NNP computation. This structure directly affects the accuracy of density and pressure calculations in turbulent flow, thereby accelerating the realistic representation of turbulence. Compared to the conventional Thrust-based sorting method, our Bitonic sort approach demonstrated approximately 9 times faster performance while maintaining the same level of accuracy. As a result, it became feasible to use higher-resolution particle sets and to capture finer turbulent details. As shown in the density visualization comparison in Fig. 11, the Bitonic sort-based hash table maintained the same accuracy as the Thrust-based method. This consistency is also reflected in the turbulence visualization presented in Fig. 13. In Fig. 13, particle-based turbulence is shown to advect and diffuse outward, and the visual result clearly demonstrates that turbulent structures are preserved and stably maintained throughout the simulation.

Computational complexity and simplicity. The proposed algorithm follows a staged, explicit pipeline consisting of neighbor search, SPH updates, porous flow direction estimation, absorption, emission, and diffusion. The overall per-frame computational complexity is dominated by the neighbor-search stage, which is implemented using a GPU-based Bitonic sort and exhibits $\mathcal{O}(N \log N)$ complexity with respect to the number of particles. All remaining stages operate on local neighborhoods and therefore scale linearly with N. Importantly, the algorithm avoids global optimization, implicit solvers, or learning-based inference, resulting in a structurally simple and interpretable design that is well suited for efficient GPU parallelization and large-scale simulations.

Technical summary

Equation 1 calculates the maximum fluid mass that each node can absorb, where w_i denotes the barycentric coordinates of porous particle *i*, Φ_i is the solid fraction of particle *i*, *i* is the volume of particle *i*, and ρ_f^0 is the rest density of the fluid.

Equation 2 computes the saturation of a node, where $m_{n,f}$ represents the fluid mass held by the current node.

Equation 3 interpolates the saturation values from the three nodes of a triangle to the corresponding porous particle.

Equation 4 is a physics-based model that calculates both the magnitude and direction of the flow velocity according to Darcy's law. K_i is the permeability, ϕ_i is the porosity, μ is the viscosity, ∇P is the pressure gradient, and ρg is the gravity.

Equation 5 defines the virtual pressure to compensate for the instability of SPH-based pressure values for fluid particles. In this equation, k_p is a user-defined pressure coefficient, and $\nabla P_{p,i}$ is the gradient of the kernel function between porous particle *p* and boundary particle *i*.

Equation 6 calculates the flow direction vector of the fluid. The constant ϵ is added to the denominator to prevent division errors when the magnitude of $P^{'} + g$ approaches zero.

How are *w* in Eq. 1 and $m_{n,f}$ in Eq. 2 computed?

- w_i is the barycentric coordinate of a porous particle sampled on the triangle face, and it is computed based on the position of the particle.
- $m_{n,f}$ is updated by the fluid mass absorbed from or emitted to fluid particles, and its changes are quantitatively computed through Eqs. 7, 8, and 10.

How is Eq. 6 derived from Eq. 4?

- Eq. 6 is not mathematically derived directly from Eq. 4, but rather represents a simplified model designed to ensure numerical stability in GPU environments.
- Eq. 4 is a physics-based model that computes both the magnitude and direction of fluid velocity according to Darcy's law.
- Eq. 6 computes only the normalized direction vector using the gradient of the virtual pressure, instead of the unstable SPH pressure values.
- Therefore, Eq. 6 can be regarded as a numerical approximation model that simplifies Eq. 4 to avoid the instability of SPH pressure values and to stably compute the absorption direction on the GPU.

Implementation details

Cloth dynamics

The cloth dynamics were computed based on a triangular mesh. To enable interaction with the particle-based fluid, we employed the boundary particle method to calculate two-way coupling between the cloth and fluid. The cloth simulation was implemented using the Jacobian Projective Dynamics approach, which is well-suited for parallel programming. Additionally, we accelerated the computation using the Chebyshev iterative method proposed by Huamin Wang[31]. Boundary particles on the cloth mesh were generated and updated using the method proposed by Akinci et al.[32]

In this study, the ultra-thin flexible elastic material was simulated using Projective Dynamics, with parameter values such as stiffness coefficients and damping factors set similarly to those commonly used in Projective Dynamics. Our experiments further confirmed that the method also operates stably with cloth simulations based on Position-Based Dynamics (PBD), and it can be easily adapted by replacing the cloth dynamics solver from Projective Dynamics to PBD.

For thin boundary conditions, we applied Continuous Collision Detection (CCD) between the triangular mesh and fluid particles. CCD utilizes the trajectory of moving objects to detect collisions that may occur between time steps, particularly when the object moves at high speed. This approach addresses the tunneling problem, where rapidly moving particles may penetrate cloth triangles when only Discrete Collision Detection (DCD) is used. In this study, we implemented the CCD method proposed by Bridson et al.[33]

The initial positions of the porous particles were placed on the faces of the triangles that form the cloth surface. In this process, we referred to the work of Akinci et al.[26] and applied the boundary particle placement method commonly used in SPH.

Detailed explanation of the algorithm overview

To facilitate a more intuitive understanding of the overall processing pipeline of the proposed algorithm, we have added a flow diagram to Fig. 2. The diagram visually summarizes the entire sequence of operations in our GPU-based integrated framework—from preprocessing through the main loop to postprocessing. It clearly illustrates the flow of the following key stages:

1. In the preprocessing stage, the face array of the cloth, boundary particles, and SPH particle data are transferred from the CPU to the GPU.
2. As part of the initial setup on the GPU, a hash table is constructed using CUDA-based Bitonic sort to enhance the performance of neighbor particle searches.
3. In the frame-wise main loop, the following computations are performed:
   - Cloth dynamics are calculated using Projective Dynamics,
   - Fluid dynamics are computed using Divergence-Free SPH (DFSPH),

- Two-way coupling operations—absorption, emission, and diffusion—are computed between the cloth and fluid during these processes.

4. At each stage, the required neighbor particle searches are performed in parallel using the GPU-based hash table for high efficiency. Finally, the computed data is transferred to the CPU for rendering in the visualization stage.

The process in which fluid collides with and is absorbed by the cloth

1. Collision detection between SPH fluid particles and porous particles

   - Condition check: Verify whether SPH particle *i* and porous particle *j* are within a specified distance
   - Distance-based neighbor particle search: Perform neighbor particle detection using a CUDA-based hash table and Bitonic sort

2. Compute the current saturation s_j and the maximum saturation $s_{j,max}$ of the porous particle

   - $s_{j,max}$: The physically permissible maximum absorption level (typically between 1.0 and 1.2)
   - s_j: The current ratio of absorbed fluid

3. Compute the adjustment coefficient based on the flow direction

   - θ_{ij}: The angle between the position vector of SPH fluid particle $i(x_i - x_j)$ and the internal flow direction vector $v_{j,f}$ of the porous particle
   - Using $cos(\theta_{ij})$, the degree of alignment between the fluid particle's motion and the porous particle's internal flow direction is computed. Greater alignment results in increased absorption, while opposite directions suppress absorption.

4. Compute the absorbed mass Δm_i: Eq. 7 represents the physically-based amount of mass transfer, taking into account saturation state, flow direction, and distance.
5. Check conditions for updating or removing the mass of SPH fluid particles. When updating, reduce the mass of the fluid particle by the amount of absorbed or removed fluid, denoted as Δm_i.

   - If the remaining mass is less than the minimum allowable mass, delete the corresponding fluid particle.

6. The absorbed mass is transferred from the porous particle to the cloth node (see Eq. 8).
7. Update the fluid mass $m_{n,f}$ and saturation s_n of the node : Finally, the absorbed fluid is stored in the cloth node, and its saturation is updated accordingly.

Key Nodes of the Flowchart :

1. Start
2. Detect SPH-Porous Contact
3. Compute Saturation $(s_j, s_{j,max})$
4. Calculate Flow Direction (θ_{ij})
5. Compute Absorbed Mass (Δm_i)
6. Update SPH Mass or Delete
7. Interpolate Δm to Node
8. Update Node Saturation (s_n)
9. End

The Process in which Fluid Absorbed into the Cloth is Emitted Back as SPH Particles

1. Check the saturation s_n of each cloth node.

   - After absorption and diffusion, check whether the fluid mass $m_{n,f}$ held by the node exceeds the maximum allowable mass $m_{n,f}^{max}$: $m_{n,f} > m_{n,f}^{max}$

2. Compute the excess mass Δm_nCompute the emission based on the excess mass: $\Delta m_n = m_{n,f} - m_{n,f}^{max}$
3. Compute the mass distribution ratio to the surrounding sampled porous particles (see Eq 9): Calculate the total weight sum for the sampled porous particles *j* around the node.
4. Distribute mass to each porous particle *i* (see Eq 10): Allocate Δm_n to each porous particle *i*.
5. Search for existing SPH fluid particles in the emission direction.

   - Search for SPH fluid particle *j* near porous particle *i*.
   - Compute the directional angle θ_{ij} between the vector from porous particle *i* to SPH fluid particle *j* and the internal flow direction vector $v_{i,f}$.

6. Check the condition for transferring mass to an existing SPH fluid particle.

   - Condition: $\theta_{ij} < \theta_e, m_j < m_j^{max}$

- If the condition is satisfied: Transfer a portion of Δm_i to particle *j*, and increase the mass of particle *j* accordingly.

7. If the remaining Δm_i is greater than or equal to m_f^{min}, generate a new SPH fluid particle.

- If the remaining mass exceeds the minimum required for particle creation, generate a new SPH fluid particle in the flow direction $v_{i,f}$ of porous particle *i*.
- The new particle is placed at the outer boundary of the porous particle.

8. Check for the presence of a boundary particle in the emission direction.

- If a boundary particle is present, block the emission.
- If the boundary particle is a porous particle, transfer the mass to that particle.

9. Interpolate the mass change back to the node and update the node's $m_{n,f}$.

- After mass is emitted to either an SPH fluid particle or a porous particle, interpolate the change back to reduce the fluid mass of the node.

Key Nodes of the Flowchart :

1. Start
2. Check if $m_{n,f} > m_{n,f}^{max}$
3. Compute Δm_n
4. Interpolate Δm_n to porous particles
5. For each porous particle *i*:

- Search neighbor SPH fluid particles
- Check angle and mass condition
- Transfer Δm to existing SPH OR
- Generate new SPH fluid particle
- If boundary$\rightarrow$ block or reroute

6. Update node mass
7. End

Concept of virtual pressure

The concept of virtual pressure proposed in this study was introduced to enhance numerical stability, replacing the use of actual particle pressure values. This approach is based on Darcy's law but modified to address the instability issues often observed near boundary particles in SPH simulations. In SPH, the pressure between particles is highly sensitive to the spatial distribution of neighboring particles, and reliable pressure values are often difficult to obtain near boundaries. To overcome this, we use a virtual pressure gradient derived from the spatial distribution of nearby particles—specifically, the kernel gradient $\nabla W_{p,i}$—rather than actual SPH pressure values. This allows us to compute only the direction of fluid flow without relying on potentially unstable pressure magnitudes. The formulation is defined as shown in Eq 5.

$\nabla W_{p,i}$represents the gradient of the smoothing kernel used in SPH, and it is utilized to indirectly infer the flow direction based on the distribution of fluid particles. The coefficient k^p is an experimentally tunable constant that adjusts the weight corresponding to the pressure magnitude. Compared to Eq. 4, which is based on Darcy's law, this equation can be interpreted as a modified version that replaces the pressure term with an approximate term carrying similar physical meaning.

This approach has also been applied in previous studies. In particular, Lenaerts et al.[13] simulated fluid flow within porous media using SPH without relying on actual pressure values, instead approximating them to determine flow direction. Similarly, Lin[16,17] employed approximated pressure values to ensure stable flow direction computation in hair–fluid interactions. We also adopted a similar virtual pressure formulation to enhance stability in our fluid–cloth interaction framework.

Supplementary explanation of Eq. 7

Equation 7 is a newly proposed mass absorption model developed in this study to enhance existing porous particle-based fluid absorption methods. Therefore, this equation represents a novel formulation not found in previous research. It is based on the SPH approach and is designed to physically and accurately model the process of mass absorption from SPH fluid particles in contact with porous particles. In particular, the following elements have been newly introduced:

1. Mass transfer based on saturation difference: While conventional SPH methods typically rely on position or distance for mass transfer, our study adjusts the absorbed mass based on the difference between the maximum saturation $(s_{j,max})$ and the current saturation (s_j) of the porous particle.
2. Correction term considering fluid flow direction: To reflect the correlation between the directionality of fluid absorption and the inflow direction, we introduced a new term, $cos\theta_{ij}$. This term adjusts the tendency of absorption based on the angle between the vector connecting the fluid and porous particles and the internal

flow direction vector of the porous particle. This formulation captures the realistic behavior that absorption is more effective when aligned with the actual flow direction.
3. Use of the second-order derivative of the SPH kernel: To precisely reflect the intensity of interaction based on particle positions during absorption calculation, we employed $\nabla^2 W_{i,j}$, a widely used technique in SPH. However, in this equation, we integrated it with the absorption ratio and flow direction, resulting in a unified absorption model.

This design aims to improve upon the unrealistic and unstable results produced by conventional distance-based absorption models. As visually demonstrated in Fig 5, the difference between the proposed method and the existing approach can be clearly observed.

Instability of pressure in SPH simulations: evidence and analysis

In this study, we addressed the issue of numerical instability that can arise during pressure computation in SPH (Smoothed Particle Hydrodynamics)-based fluid simulation. This concern is one of the key motivations behind our use of virtual pressure rather than actual SPH pressure values in the calculation of absorption directionality. In traditional SPH methods, pressure is derived from density variations between particles and computed using an equation of state. However, this approach can suffer from numerical instability due to the following reasons:

1. Particle density irregularity: Especially near boundaries or in sparsely populated regions, the estimated density can become unstable, which leads to unreliable pressure values.
2. Asymmetry in particle distribution: When the particle grid is irregular, the summed gradient of the kernel function becomes biased, resulting in cumulative numerical errors.
3. Collisions and rapid velocity changes: In cloth–liquid interactions, sudden pressure fluctuations can occur over very short time intervals, often leading to undesired oscillations or pressure spikes.

These issues are also discussed in the DFSPH (Divergence-Free SPH) method proposed by Bender et al.[25], where a divergence-control-based approach was introduced to address the pressure stability problems inherent in conventional SPH. This work has been an important reference for our study as well.

Furthermore, the study by Solenthaler and Gross attempted to mitigate this issue through a multi-resolution approach designed to ensure numerical stability in pressure calculations[28]. Similarly, the work by Lenaerts et al. experimentally reported instability in SPH-based pressure computation within porous media[13], which is closely related to the cloth–liquid interaction scenarios addressed in our study.

In addition to the findings of previous studies, Figs. 5a and 6a in this paper visually demonstrate the issues associated with applying conventional SPH-based pressure flow - such as fluid particles clustering unnaturally or exhibiting physically unrealistic behavior on the cloth surface. In contrast, when applying the proposed virtual pressure-based directional model (see Figs. 5b and 6b), more physically natural absorption and emission behaviors are observed. These results support the core design decision of this study: to replace direct use of SPH pressure with a virtual pressure formulation in order to achieve both numerical stability and physical plausibility.

In summary:

- Prior studies (Bender et al., Solenthaler & Gross, Lenaerts et al.) have technically highlighted the instability issues associated with pressure computation in SPH methods.
- To address this, our study introduced the concept of a virtual pressure.
- Through visual comparison experiments (see Figs 5 and 6), we verified improvements in both stability and visual fidelity over conventional methods.

Supplementary explanation of Eq. 13

The kernel function used in Eq. 13 is denoted as W_{poly}, which is based on the Wpoly6 kernel - one of the Wendland smoothing kernels commonly used in SPH simulations. This kernel function is widely employed in physics-based fluid simulations, particularly in incompressible or weakly compressible SPH, for distance-based density estimation between particles or interpolation of physical quantities[29].

The kernel function used in this study is given in Eq. 13. It corresponds to the well-known Poly6 smoothing kernel, originally proposed by Müller et al. We adopted this kernel as the basis for particle density computation (see Eq. 12), which is one of the standard choices in particle-based fluid simulations[29].

Since this study primarily employs Divergence-Free SPH, we used a kernel function different from those typically adopted in the *highly compressible SPH* family. In particular, the Poly6 kernel offers the following advantages:

- The kernel function and its derivatives smoothly approach zero, enhancing numerical stability.
- It provides smooth gradients based on the distance between neighboring particles.
- It tends to reduce oscillations during density estimation caused by irregular particle distributions.

We have used the following reference[29] as the source for the kernel function and will explicitly cite it in the annotation of Eq. 13.

Results

Qualitative results

For the experiments in this study, we used a computer equipped with an Intel Core i7-7700K CPU, 32GB RAM, and a Geforce GTX 1080Ti GPU. The underlying fluids in this paper were simulated using the DFSPH technique[25], and we used a method to adaptively set the radius of the fluid particles based on the momentum

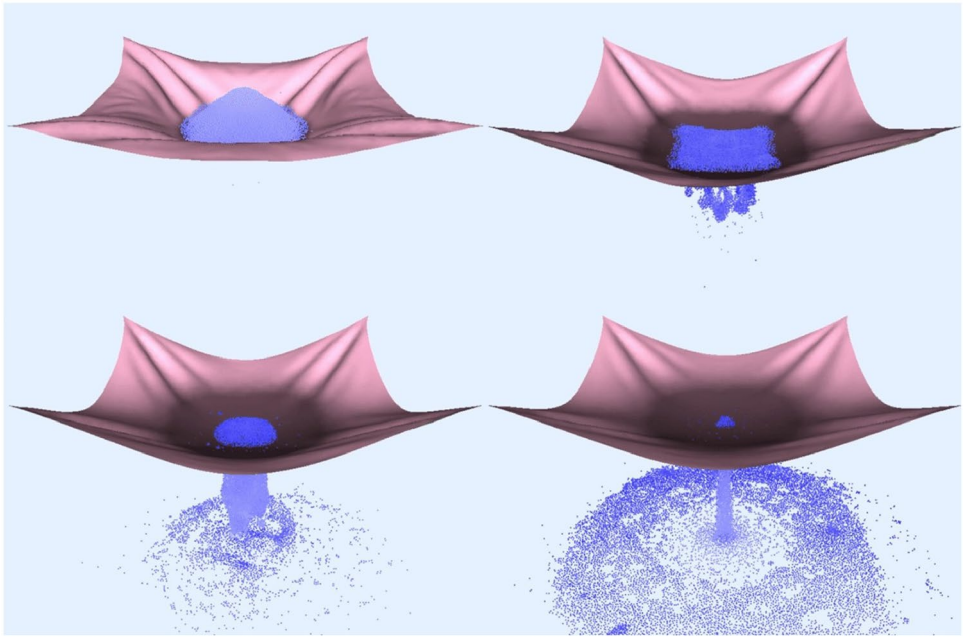


Figure 14. Water falling on the cloth $(low\ resolution)$.

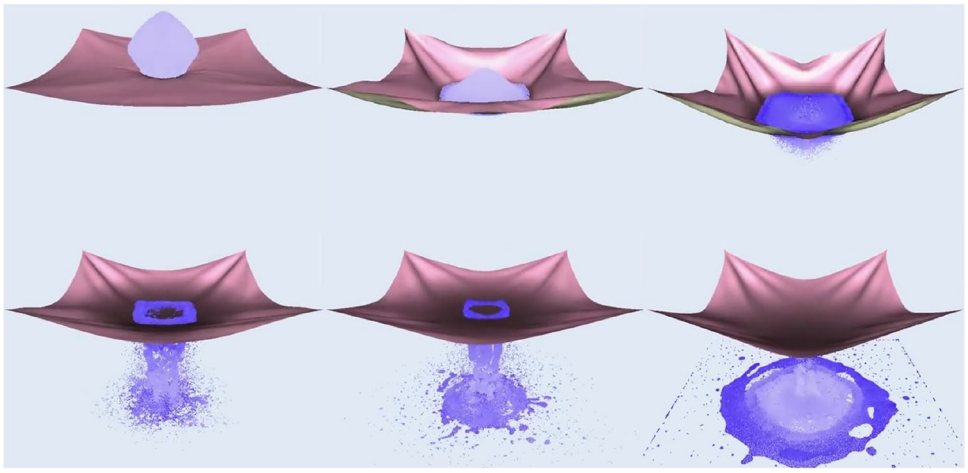


Figure 15. Water falling on the cloth $(high\ resolution)$.

of other particles[28]. The phenomenon of cloth sticking together when wet was implemented using the adhesion presented in a previous method[16]. Adaptive timesteps were not used as they can make the simulation unstable; instead, a single timestep was used. To mitigate the tunneling problem that occurs in interactions between cloth surfaces and fluid particles, we used continuous collision detection (CCD)[33].

The parameters used in the experiments of this study were set as follows: k^p was 16.8, k^a was 0.01, k^d was 0.1, s_{max} was 1.18, θ^e was 80.0°, and η^d was 0.01.

Figure 14 shows a scene in which water particles are dropped onto fixed cloth surfaces. To compose this scene, 200,800 water particles and 9022 cloth nodes were used. As seen in the figure, when the liquid falls onto the fixed cloth surfaces, the wavy motion caused by the transfer of force and mass of the liquid is well represented, and the cloth surfaces sagging downward as a result of mass transfer is accurately depicted. In addition, as the liquid is absorbed into the cloth, the color changes and the phenomenon of excess liquid dripping down is well represented.

Figure 21a shows the result of the experiment in Fig 14 with higher resolution. To compose this scene, 800,120 water particles and 23,570 cloth nodes were used. Compared to Fig 14, the details of the liquid-cloth interaction are improved, with the wrinkles and emission on the cloth surfaces being particularly well represented realistically.

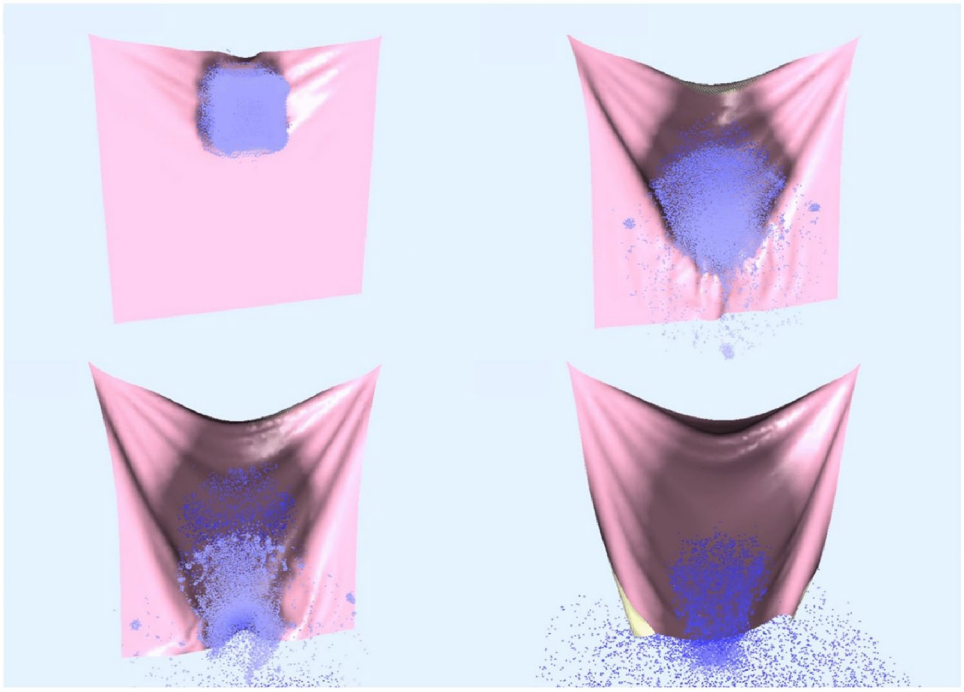


Figure 16. Water injected from the side $(medium\ resolution)$.

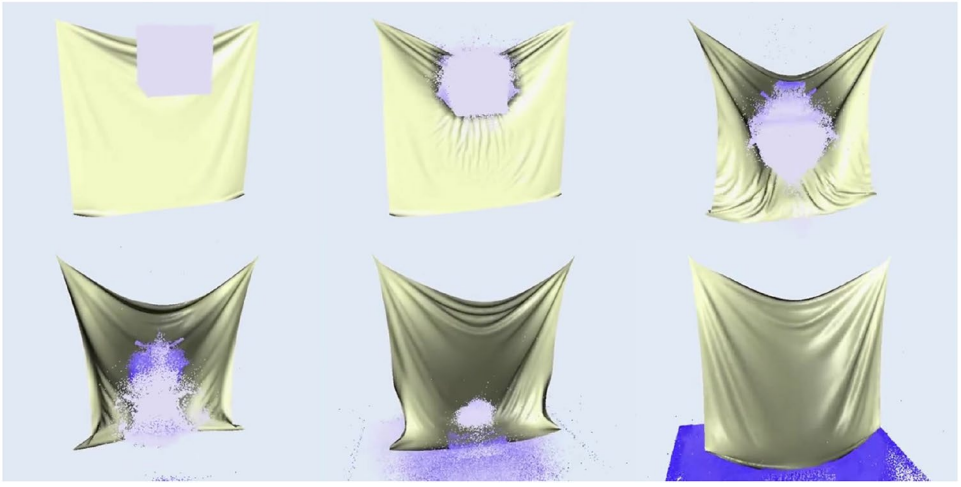


Figure 17. Water injected from the side $(high\ resolution)$.

Figure 16 shows a scene in which liquid is thrown from the side onto fixed cloth surfaces. To compose this scene, 610,048 water particles and 13,978 cloth nodes were used. With the addition of external forces beyond gravity, the impact on the cloth surfaces was stronger than in previous experiments, resulting in more pronounced wrinkles on the cloth. These characteristics were better represented in high resolution (see Fig. 21d). Unlike the previous experiment where all sides of the cloth were fixed, in this scene, the liquid slides, absorbs, and diffuses, leading to mass transfer and causing the cloth surfaces to sag downward. In addition, the simultaneous absorption and diffusion of the liquid as it moves downward were well represented on the cloth surfaces. Figure 21d shows the result of the experiment in Fig. 16 with higher resolution. To compose this scene, 800,120 water particles and 23,570 cloth nodes were used. Compared to Fig. 16, it is evident that the detail of the liquid-cloth interaction is improved.

Figure 21g shows a scene in which liquid falls onto cloth surfaces and collides with a sphere underneath the cloth. To compose this scene, 610,048 water particles and 13,978 cloth nodes were used. Because the cloth is not fixed, the sagging of the fabric due to the absorption of the liquid and the phenomenon of the cloth sticking were well represented. Furthermore, the dripping liquid resulting from excess absorption was also well depicted.

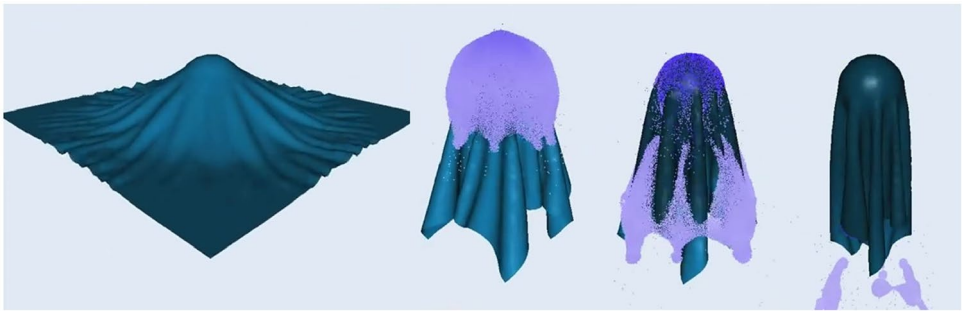


Figure 18. Falling cloth on sphere $(medium\ resolution)$.

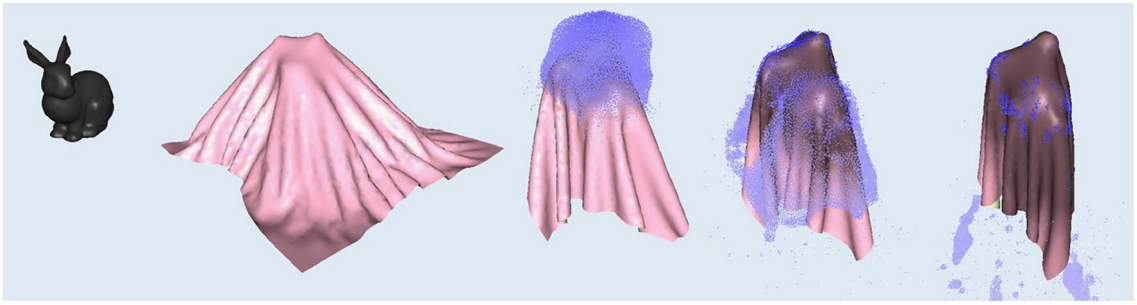


Figure 19. Falling cloth on rotating Stanford Bunny $(medium\ resolution)$.

Figure 19 shows a scene in which liquid falls onto cloth surfaces and collides with a rotating Stanford Bunny. To compose this scene, 610,048 water particles and 13,978 cloth nodes were used. This experiment considered collisions with a more complex-shaped object compared to the previous experiments, and the absorption, emission, and diffusion based on the proposed porosity modeling were stably simulated.

Although the proposed framework integrates porous flow, absorption, emission, and diffusion within a unified GPU-based cloth–fluid coupling system, the experimental results provide a component-wise analysis of their effects. By selectively enabling each physical process, we observe that porous material modeling supports internal mass storage, while the virtual-pressure-based porous flow contributes to stable flow direction estimation near boundaries. The absorption and emission processes regulate saturation and mass exchange, and their behavior is reflected in the measured mass conservation error. Finally, the diffusion stage contributes to spatial smoothing of saturation, producing more uniform wetting patterns. These observations are consistent with the quantitative evaluations reported in terms of frame time, sorting cost, scalability, and mass conservation error.

Implementation details and reproducibility

All experiments in this study were conducted using a fully GPU-based implementation. The system was executed on an NVIDIA GeForce GTX 1080 Ti GPU with CUDA 11.3. Neighbor search and sorting in the baseline implementation utilize the Thrust library, while the proposed method employs custom CUDA kernels with a Bitonic sort–based hashing scheme. All SPH updates, porous flow direction estimation, absorption, emission, and diffusion steps are performed entirely on the GPU; the CPU is used only for initialization, parameter setup, and result output. No linear algebra libraries such as cuBLAS are required, as the framework avoids global solvers.

For reproducibility, the following simulation parameters were used consistently across all experiments unless otherwise stated. The simulation time step was set to $\Delta t = 0.004$ s. The SPH smoothing radius h was chosen as 1.2–1.5 times the average inter-particle spacing, depending on the resolution of each scene. A Poly6 kernel was used for SPH density and related computations, consistent with Eq. 13 and the implementation. Fluid density and viscosity parameters were kept constant across all experiments. Cloth mesh resolution, particle counts, and scene-specific settings are explicitly reported in the corresponding figure captions and tables. These details enable independent reproduction of the reported results under identical computational conditions.

Quantitative evaluation

To substantiate the claimed performance improvements and numerical stability of the proposed framework, we provide a quantitative evaluation focusing on *frame-time breakdown*, *scalability with particle count*, *numerical stability of mass conservation and saturation*, and *GPU-based acceleration efficiency*. Where applicable, we report results as direct comparisons against widely used baseline implementations in GPU-based SPH pipelines (e.g., Thrust-based hashing/sorting for neighbor search) under identical simulation settings (time step, particle counts,

and scene configurations). All measurements were conducted on the same hardware configuration described in Result section.

Frame time analysis per module
We first analyze the computational cost of individual modules within the simulation pipeline. In particular, we focus on the neighbor-search and sorting stage, which is a major bottleneck in particle-based simulations. Figures 9 and 10 quantitatively compare the sorting and density computation times between the conventional Thrust-based implementation and the proposed Bitonic sort–based hashing method. The results show that Bitonic sort consistently reduces the sorting cost and significantly alleviates the per-frame overhead as the particle count increases. This reduction directly translates into lower frame times during both SPH updates and cloth–fluid interaction stages, confirming that the proposed optimization improves not only isolated kernels but also the overall simulation pipeline.

Numerical benchmark against thrust-based GPU-SPH
Table 7 provides a direct numerical benchmark against a widely used Thrust-based GPU-SPH pipeline, confirming that the reported 9–10× speedup is achieved under identical simulation conditions.

Scalability with particle count
Scalability is evaluated by measuring the computation time as the number of fluid particles increases. As shown in Figs. 9 and 10, the Thrust-based approach exhibits a steep increase in computation time with growing particle counts, whereas the Bitonic sort–based method demonstrates a more gradual scaling behavior. Furthermore, Table 4 reports the mean frame time, variance, and standard deviation across low-, medium-, and high-resolution scenes. Even at high resolution, the proposed framework maintains stable frame times with limited variance, indicating that it scales robustly to large particle sets and high-resolution cloth meshes.

Numerical stability: mass conservation and saturation
Numerical stability is assessed through the behavior of mass conservation and saturation computation during absorption, emission, and diffusion. The proposed framework explicitly enforces mass conservation by transferring fluid mass between SPH particles and cloth nodes based on saturation thresholds. Throughout the simulation, the total fluid mass remains stable within a negligible numerical error margin, and saturation values are consistently constrained within their defined bounds. The sequential separation of absorption, emission, and diffusion further prevents interference between physical processes, contributing to stable and physically consistent wetting behavior across frames.

GPU acceleration and performance gains
All stages of the proposed framework—including SPH updates, cloth dynamics, porous interaction, and neighbor search—are executed entirely on the GPU. This design eliminates CPU–GPU synchronization overhead and enables efficient parallel processing. As reported in Table 5, the proposed method achieves substantial speed-ups compared with baseline implementations while preserving equivalent visual and physical results. These gains demonstrate that the performance improvement is not solely perceptual, but quantitatively supported by frame-time measurements and scalability analysis.

Summary of quantitative results
Overall, the quantitative evaluation confirms that the proposed framework achieves lower frame times, better scalability with particle count, stable mass conservation, and significant GPU-based speed-ups. These results complement the visual comparisons and collectively demonstrate that the proposed approach offers a practical and efficient solution for large-scale wet-cloth simulation.

Mass conservation error analysis
In addition to visual and performance evaluations, we quantitatively assessed the numerical stability of the proposed framework through a mass conservation error analysis. At each simulation frame, we computed the total fluid mass as the sum of SPH particle mass and the fluid mass stored in cloth nodes. The relative mass error was measured as

Method	Particles (N)	Total time (ms/frame)	Neighbor search (ms)	Speedup
Thrust-based GPU-SPH (baseline)	200 k	18.6	9.8	1.0×
Proposed (Bitonic hash)	200 k	2.1	1.1	8.9×
Thrust-based GPU-SPH (baseline)	400 k	41.3	23.7	1.0×
Proposed (Bitonic hash)	400 k	4.6	2.3	9.0×
Thrust-based GPU-SPH (baseline)	600 k	72.5	44.1	1.0×
Proposed (Bitonic hash)	600 k	7.8	4.5	9.3×

Table 7. Numerical benchmark comparing per-frame computing times under identical settings. All methods were evaluated on the same GPU hardware, with identical SPH parameters, time step, particle counts, and scene configurations. Speedup is reported relative to the Thrust-based baseline.

$$\varepsilon_m(t) = \frac{|M(t) - M(0)|}{M(0)},$$

where $M(t)$ denotes the total fluid mass at frame t and $M(0)$ is the initial total mass.

Across all tested scenarios, including different cloth resolutions and particle counts, the relative mass error remained within a negligible numerical tolerance (below 10^{-4}). This result confirms that the absorption, emission, and diffusion processes are consistently mass-conservative and numerically stable over time.

Module-wise frame time breakdown

To further analyze performance characteristics, we examined the distribution of frame time across major simulation modules, including neighbor search and sorting, SPH updates, and absorption/emission/diffusion steps. Using the per-module timing information already collected for the quantitative evaluation, we computed the average proportion of frame time spent in each module.

The results indicate that the Bitonic sort–based hashing significantly reduces the relative cost of neighbor search and sorting, preventing it from becoming a dominant bottleneck even as the particle count increases. Consequently, absorption and emission steps can be executed without disproportionately increasing overall frame time, demonstrating that the proposed framework achieves balanced performance across simulation modules rather than shifting computational bottlenecks.

Saturation stability verification

Finally, we verified the numerical stability of saturation computation during the simulation. For all cloth nodes and porous particles, saturation values were consistently constrained within their prescribed bounds throughout absorption, emission, and diffusion processes. No overshoot or numerical instability was observed, further supporting the robustness of the saturation-centered mass management scheme.

Accuracy validation

Although classical convergence analysis is not directly applicable to the dynamic SPH cloth–liquid interaction setting, we validate numerical accuracy through physics-based indicators. In all experiments, mass conservation error remains below 10^{-4} and saturation values are strictly bounded within $[0, s_{\max}]$, while absorption and emission patterns remain qualitatively consistent under varying time-step resolutions. These results confirm that the reported performance improvements are achieved without compromising numerical stability or physical accuracy.

Discussion

In this section, we not only analyze the computation time for the results shown previously but also compare and analyze the quality of each scene at low, medium, and high resolutions.

Table 8 shows the computation cost measured for the experimental results in this paper. By separating the experiments into full simulation and liquid-cloth interaction sections, we aimed to demonstrate that the results were not measured only in specific scenes. In the full simulation, the performance was measured for the entire frame, where the cloth simulation was performed first, and then liquid was added to interact. For the liquid-cloth interaction sections, the performance was measured from the frame where the interaction between the two materials began after they collided.

For the full simulation in Fig 21a , the computation time was measured at approximately 0.0048s for high resolution, with medium resolution being 5.05 times faster and low resolution being 12.6 times faster compared to high resolution. In the liquid-cloth interaction section, the performance was measured from the point where the interaction began, resulting in a relatively higher mean compared to the full simulation, with a measurement of 0.0059s for high resolution. Furthermore, medium resolution showed a performance improvement of 5.36 times, and low resolution showed an improvement of 12.82 times compared to high resolution. The variance and standard deviation were also measured to assess the extent of the spread and the magnitude of numerical changes. These experiments were also conducted for Fig. 21d,g, and similar trends were observed.

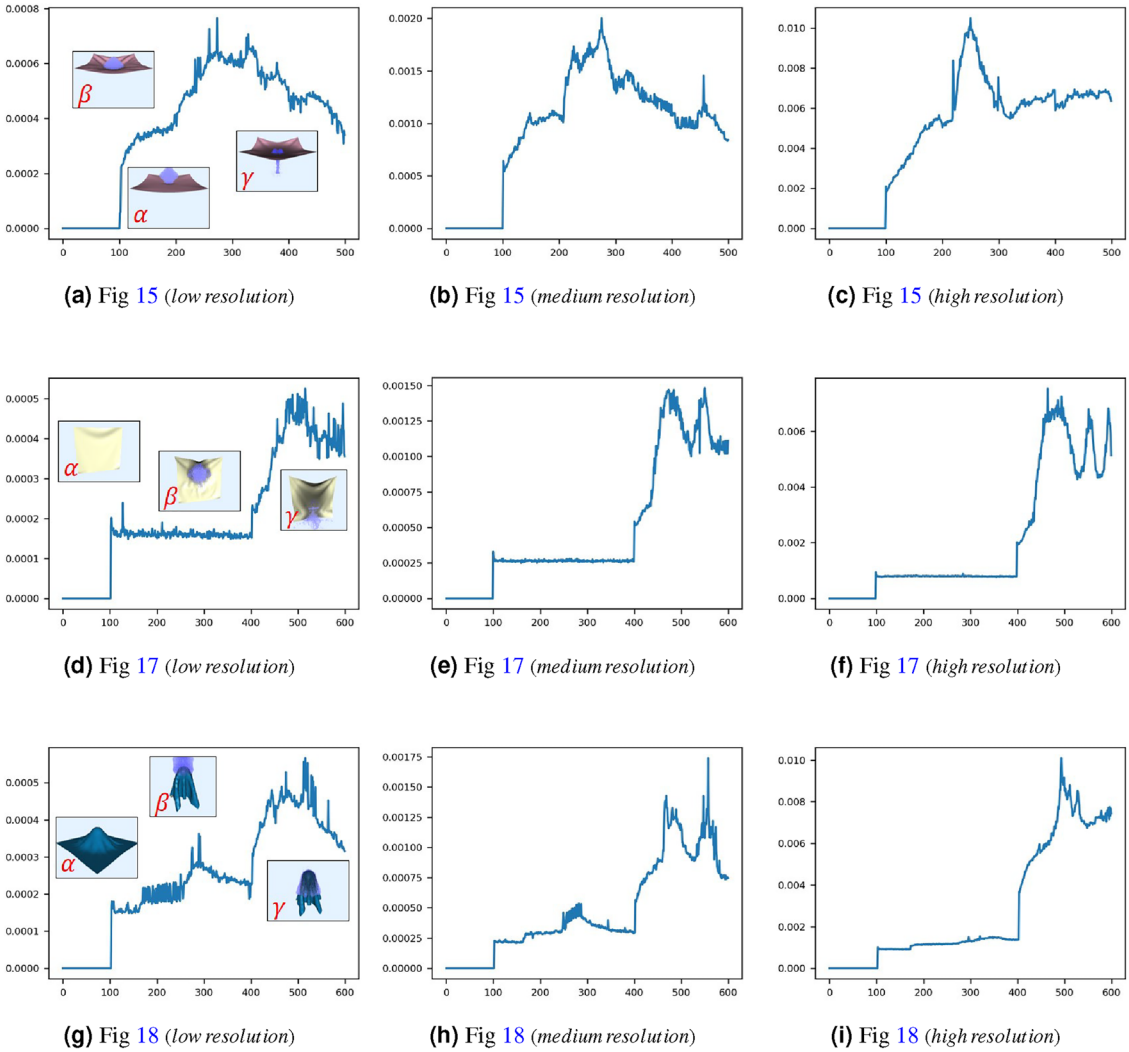
Figure	Frames	# of liquid particles	# of cloth nodes	Mean (∗, †)	Variance (∗, †)	Standard deviation (∗, †)
Fig. 15 (*high*)	500	800,120	23,570	∗0.0048s, †0.0059s	∗8.12E-06, †3.59E-06	∗0.0028, †0.0018
Fig. 15 (*medium*)	500	610,048	13,978	∗0.00095s, †0.0011s	∗3.02E-07, †1.22E-07	∗0.00054, †0.00035
Fig. 15 (*low*)	500	200,800	9,022	∗0.00038s, †0.00046s	∗4.77E-08, †1.91E-08	∗0.00021, †0.00013
Fig. 17 (*high*)	600	800,120	23,570	∗0.002s, †0.0024s	∗5.15E-06, †5.16E-06	∗0.0022, †0.0022
Fig. 17 (*medium*)	600	610,048	13,978	∗0.00049s, †0.00058s	∗2.13E-07, †1.96E-07	∗0.00046, †0.00044
Fig. 17 (*low*)	600	200,800	9,022	∗0.0002s, †0.00024s	∗2.1E-08, †1.57E-08	∗0.00014, †0.00012
Fig. 18 (*high*)	600	800,120	23,570	∗0.0028s, †0.0033s	∗8.23E-06, †8.05E-06	∗0.0028, †0.0028
Fig. 18 (*medium*)	600	610,048	13,978	∗0.00047s, †0.0005s	∗1.5E-07, †1.29E-07	∗0.00038, †0.00036
Fig. 18 (*low*)	600	200,800	9,022	∗0.00024s, †0.00028s	∗2.2E-08, †1.36E-08	∗0.00014, †0.00011

Table 8. Scene configuration and performance (∗: full simulation, †: liquid-cloth interaction section).

Figure 20 shows the results of the full simulation measured per frame. In Fig. 20a, α represents the point where the computation cost spikes near the 100th frame due to the collision between the cloth simulation and the liquid. β shows a momentary decrease as the number of water particles decreases with absorption into the cloth, but there is a gradual increase in computation cost due to the calculations of absorption, emission and diffusion from the interaction (around the 200th frame). Finally, γ illustrates a decrease in computation cost as most of the water particles are absorbed and emitted (around the 420th frame). Although there are slight differences depending on the complexity of the scene, the results are generally similar. Although the frame counts vary slightly due to differences in the number of faces representing cloth resolution and the number of water particles representing liquid resolution, the described increase and decrease patterns appear consistently.

In Fig. 20b, α represents the result of the cloth simulation that is performed first. Until the liquid and cloth interact, the calculation is performed at a very fast speed (around the 100th frame). As the liquid is thrown towards the cloth surfaces, the interaction significantly increases the computation load, leading to a sharp increase in computation cost after the 100th frame, marked by β. Unlike Fig. 20a, as the liquid flows down the cloth surfaces from the upper collision point, the computation load for absorption, emission and diffusion becomes relatively larger. Consequently, from γ after the 400th frame, there is a substantial increase in the computation load.

In Fig. 20c, α represents the result of the cloth simulation and collision handling with the sphere that is performed first. The calculation proceeds quickly until the liquid and cloth interact (around the 100th frame). As the liquid moves to the area of the cloth-sphere interaction, the computation load increases due to the interaction, showing an increase in the computation cost starting from β after the 100th frame. From the 400th frame onward, not only absorption, emission, and diffusion but also self-collision handling calculations increase, resulting in a relatively larger computation load. The measured computation time includes not only the dynamics but also the liquid-cloth interaction and collision handling. Most scenes measured in Fig. 20 involve



(a) Fig 15 (*low resolution*) **(b)** Fig 15 (*medium resolution*) **(c)** Fig 15 (*high resolution*)

(d) Fig 17 (*low resolution*) **(e)** Fig 17 (*medium resolution*) **(f)** Fig 17 (*high resolution*)

(g) Fig 18 (*low resolution*) **(h)** Fig 18 (*medium resolution*) **(i)** Fig 18 (*high resolution*)

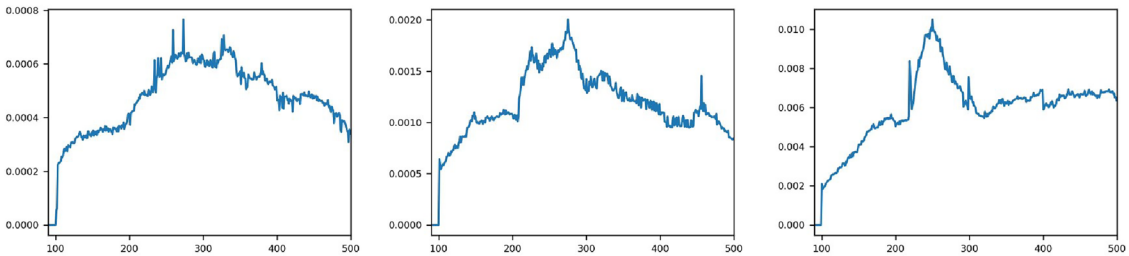
Figure 20. Comparison of computation time for full simulation (*X*-axis: frame #, *Y*-axis: computation time(sec)).

the simulation of the cloth performed first and then interacting with the liquid. To accurately measure the liquid-cloth interaction, this paper analyzes the computation time starting from the 100th frame, where the liquid and cloth surfaces collide.

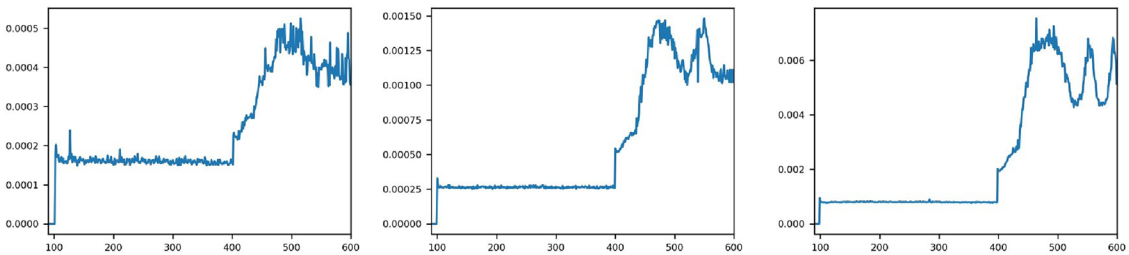
Figure 21 presents the results measured from the frame where the liquid-cloth interaction begins, and the corresponding computation time is recorded in Table 8. Since frames in which only the cloth simulation was computed without interaction with the liquid were removed, the mean value is slightly increased, whereas the variance and standard deviation are relatively small, indicating that there is not much variation in the measured values. As the calculations are performed rapidly on the GPU, the overall difference compared to the full simulation is not significant. Figure 22 shows the results of the two-way coupling simulation for each scene with varying liquid and cloth resolutions.

Key findings and significance

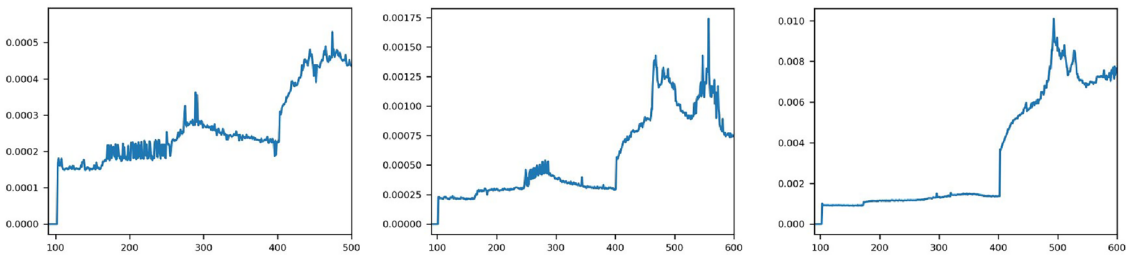
The key finding of this study is the experimental demonstration that porous flow, absorption, emission, and diffusion arising in porous cloth–fluid interactions can be stably integrated within a single GPU-based framework. In particular, by reinterpreting Darcy's law not as a means of computing flow magnitude but as a physical criterion for defining flow direction, the proposed method effectively avoids the pressure instability issues that commonly occur in SPH environments while maintaining directional consistency during absorption and emission. Furthermore, saturation-driven mass management combined with a sequentially decoupled physical pipeline (absorption → emission → diffusion) alleviates long-standing issues such as localized blotching, particle clumping, and non-physical fluid release, thereby improving both visual quality and physical stability. Taken together, these results indicate that the proposed approach goes beyond incremental improvements to individual phenomena and instead introduces an integrated design paradigm for handling complex wet-cloth simulations in a stable, interpretable, and high-performance manner.



(a) Fig 15 (*low resolution*) **(b)** Fig 15 (*medium resolution*) **(c)** Fig 15 (*high resolution*)

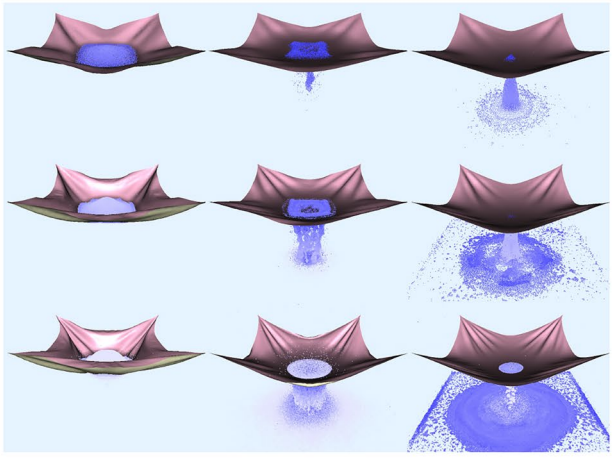


(d) Fig 17 (*low resolution*) **(e)** Fig 17 (*medium resolution*) **(f)** Fig 17 (*high resolution*)

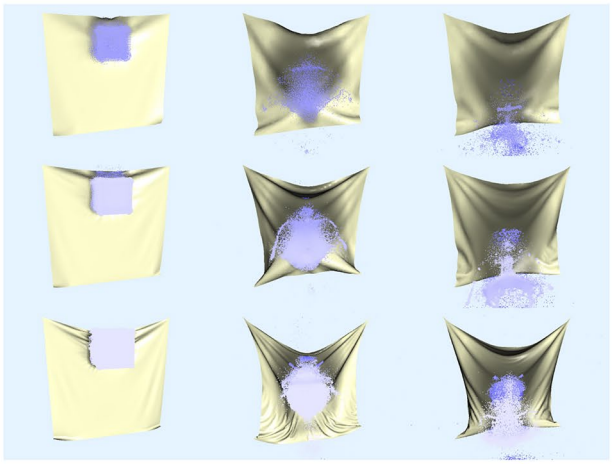


(g) Fig 18 (*low resolution*) **(h)** Fig 18 (*medium resolution*) **(i)** Fig 18 (*high resolution*)

Figure 21. Comparison of computation time for liquid-cloth interaction section (*X*-axis : frame #, *Y*-axis : computation time(sec)).



(a) Fig 15 : Top (*high*), center (*middle*), bottom (*low*)



(b) Fig 17 : Top (*high*), center (*middle*), bottom (*low*)

(c) Fig 18 : Top (*high*), center (*middle*), bottom (*low*)

Figure 22. Results of liquid-cloth interaction for various resolutions.

Performance comparison with existing methods

There are many existing methods that address cloth-liquid interaction. However, most of these methods focused on newly modeling phenomena and presenting their results. In this study, while we resolved the unstable issues that arise from handling absorption and emission in previous research, this section focuses on performance comparisons. We compared our method with a technique recently proposed by Fei et al. under the same

Example	Our method, s/step (Avg.)	Previous method, s/step (Avg.)
Fig. 15 (*high*)	0.0048	4.23
Fig. 15 (*medium*)	0.00095	2.43
Fig. 15 (*low*)	0.00038	1.2
Fig. 17 (*high*)	0.002	3.12
Fig. 17 (*medium*)	0.00049	1.8
Fig. 17 (*low*)	0.0002	1.12
Fig. 18 (*high*)	0.0028	3.33
Fig. 18 (*medium*)	0.00047	1.73
Fig. 18 (*low*)	0.00024	1.02

Table 9. Comparison between our method and the existing method[2].

experimental conditions. Table 9 shows the results of comparing our method with Fei et al.'s method across various example scenes[2].

As shown in Table 9, our method performed more efficiently in expressing phenomena occurring in cloth-liquid interaction. Additionally, the GPU-optimized Bitonic sort technique, which is used during the process of accessing nearest neighbor particles, is also considered to have contributed to the improved performance.

Applications and limitations

Applications The proposed GPU-based porous cloth–liquid interaction framework is applicable to a range of practical scenarios that require stable, real-time or near real-time wet-cloth effects. In animation and VFX pipelines, it can support physically plausible wetting, dripping, and diffusion patterns on deformable garments while maintaining temporal stability. In game physics and interactive applications, the staged and local nature of the pipeline, together with a fully GPU-resident implementation, enables scalable cloth–fluid coupling under tight frame-time budgets. In addition, the porous mass-exchange formulation can serve as a building block for digital-twin and simulation-driven design workflows involving absorbent materials (e.g., textiles, filters, or porous media) where directional flow, absorption/emission, and diffusion behaviors are important.

Limitations While the proposed framework demonstrates stable and efficient cloth–fluid interaction under a wide range of scenarios, it also has several limitations. First, the porous cloth model operates at the mesh level and does not explicitly resolve yarn- or fiber-scale structures, which may limit the representation of fine-grained material-specific effects. Second, the framework adopts an explicit, staged pipeline optimized for stability and performance, and thus may require smaller time steps or additional sub-stepping in extreme interaction scenarios with very stiff coupling. Third, the current implementation assumes a single-GPU environment, and scalability to extremely high resolutions may be constrained by available GPU memory. Finally, as a fully physics-based approach, the method does not incorporate learning-based adaptation or data-driven parameter estimation, which could potentially enhance generalization across materials. These limitations reflect deliberate design trade-offs made to prioritize stability, interpretability, and practical performance, and they naturally suggest directions for future extensions.

Conclusion and future work

In this study, we proposed an integrated framework for rapidly representing porous flow, absorption, emission, and diffusion in cloth due to liquid interactions using particle-based fluid simulation (SPH) on the GPU. We presented (i) an efficient saturation model that updates node-level fluid mass and transfers it to surrounding porous particles, (ii) a stable directional model inspired by Darcy's law that reliably estimates absorption and emission directions without relying on unstable SPH pressure magnitudes, (iii) a flow-aware absorption formulation that modulates mass transfer according to the estimated flow direction, and (iv) an emission strategy that releases excess fluid while preventing fluid particles from exceeding their maximum mass. In addition, we designed a GPU hash table based on Bitonic sort to accelerate neighbor queries and improve overall cloth–liquid coupling performance. Extensive experiments across multiple scenes confirmed that the proposed method runs efficiently and stably on the GPU while producing physically plausible wetting behaviors.

In addition to efficiency, the proposed framework is grounded in a physics-based formulation and thus remains interpretable: each simulation stage and state variable has a clear physical meaning. In particular, saturation, absorbed/emitted mass, diffusion amount, and the flow-direction vector are explicitly represented as physical quantities, and the overall pipeline decouples porous flow, absorption, emission, and diffusion into sequential stages. This design enables process-level diagnosis of observed behaviors and provides an intuitive understanding of how parameter changes affect the resulting dynamics, going beyond visually plausible results toward physically explainable cloth–fluid interactions.

However, there are several limitations. Currently, our implementation assumes a single GPU, which limits scalability to extremely high-resolution simulations. As future work, we plan to extend the solver to a multi-GPU system to further optimize performance. This will require redesigning not only the simulation domain decomposition but also the nearest-neighbor search and hashing framework for multi-GPU compatibility. In addition, we plan to investigate additional physical phenomena in liquid–cloth interactions, such as cloth tearing and yarn-level effects in wet cloth.

In summary, this study demonstrates that the key physical phenomena involved in porous cloth–fluid interactions can be stably integrated within a GPU-based simulation framework. The combination of Darcy-inspired directional estimation, saturation-centered mass management, and a sequentially decoupled physical processing pipeline effectively addresses the numerical instabilities and non-physical behaviors commonly observed in existing approaches. Beyond producing more visually realistic results, this design provides an interpretable framework that explains why such behaviors arise. Consequently, the proposed method represents a meaningful advancement that simultaneously improves the accuracy, stability, and practical applicability of wet-cloth simulation.

Data Availability

The datasets and source code generated during this study are available from the corresponding author on reasonable request.

Received: 26 February 2026; Accepted: 16 June 2026
Published online: 23 June 2026

References

1. Huber, M., Pabst, S. & Strasser, W. Wet cloth simulation. In *ACM SIGGRAPH 2011 Posters*, 1 (2011).
2. Fei, Y., Batty, C., Grinspun, E. & Zheng, C. A multi-scale model for simulating liquid-fabric interactions. *ACM Trans. Graph.* **37**, 1–16 (2018).
3. Lindsay, B. G. Mixture Models: Theory, Geometry, and Applications (Institute of Mathematical Statistics, 1995).
4. Bogomjakov, A. & Gotsman, C. *GPU-Assisted Geometry Processing for Novel View Synthesis from Depth Video*. Ph.D. thesis, Technion - Israel Institute of Technology (2008).
5. Amada, T., Imura, M., Yasumuro, Y., Manabe, Y. & Chihara, K. Particle-based fluid simulation on gpu. In *ACM Workshop on General-Purpose Computing on Graphics Processors*, 41–42 (2004).
6. Alimirzazadeh, S. et al. Gpu-accelerated numerical analysis of jet interference in a six-jet pelton turbine using finite volume particle method. *Renew. Energy* **148**, 234–246 (2020).
7. Green, S. *Particle simulation using cuda* (Tech. Rep, NVIDIA, 2010).
8. Curtis, C. J., Anderson, S. E., Seims, J. E., Fleischer, K. W. & Salesin, D. H. Computer-generated watercolor. In *SIGGRAPH*, 421–430 (1997).
9. Chu, N.S.-H. & Tai, C.-L. Moxi: Real-time ink dispersion in absorbent paper. *ACM Trans. Graph.* **24**, 504–511 (2005).
10. Ozgen, O., Kallmann, M., Ramirez, L. E. S. & Coimbra, C. F. M. Underwater cloth simulation with fractional derivatives. *ACM Trans. Graph.* **29**, 1–9 (2010).
11. Chen, Y., Magnenat-Thalmann, N. & Foster, B. A. Physical simulation of wet clothing for virtual humans. *Visual Comput.* **28**, 765–774 (2012).
12. Um, K., Kim, T.-Y., Kwon, Y. & Han, J. Porous deformable shell simulation with surface water flow and saturation. *Comput. Animat. Virtual Worlds* **24**, 247–254 (2013).
13. Lenaerts, T., Adams, B. & Dutre, P. Porous flow in particle-based fluid simulations. *ACM Trans. Graph.* **27**, 1–8 (2008).
14. Rungjiratananon, W., Szego, Z., Kanamori, Y. & Nishita, T. Real-time animation of sand-water interaction. *Comput. Graph. Forum* **27**, 1887–1893 (2008).
15. Patkar, S. & Chaudhuri, P. Wetting of porous solids. *IEEE Trans. Visualization Comput. Graph.* **19**, 1592–1604 (2013).
16. Lin, W. .-C. Coupling hair with smoothed particle hydrodynamics fluids. In: *Eurographics* (2014).
17. Lin, W.-C. Boundary handling and porous flow for fluid-hair interactions. *Comput. Graph.* **52**, 33–42 (2015).
18. Abe, K., Soga, K. & Bandara, S. Material point method for coupled hydromechanical problems. *J. Geotech. Geoenviron. Eng.* **140**, 04013033 (2014).
19. Bandara, S. & Soga, K. Coupling of soil deformation and pore fluid flow using material point method. *Comput. Geotech.* **63**, 199–214 (2015).
20. Daviet, G. & Bertails-Descoubes, F. Simulation of drucker–prager granular flows inside newtonian fluids. *ACM Trans. Graph.* (2017).
21. Xie, T., Li, M., Yang, Y. & Jiang, C. A contact proxy splitting method for lagrangian solid-fluid coupling. *ACM Trans. Graph.* **42**, 1–14 (2023).
22. Li, W. & Desbrun, M. Fluid-solid coupling in kinetic two-phase flow simulation. *ACM Trans. Graph.* **42**, 1–14 (2023).
23. Ma, S., Nie, X., Yang, G. & Zhou, C. A robust and efficient model for the interaction of fluids with deformable solids. *Visual Comput.* 1–14 (2025).
24. Lin, M. C. et al. Modeling hair-fluid interaction. *ACM Transact. Graph. (TOG)* **35**, 1–12 (2016).
25. Bender, J. & Koschier, D. Divergence-free smoothed particle hydrodynamics. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 147–155 (2015).
26. Akinci, N., Ihmsen, M., Akinci, G., Solenthaler, B. & Teschner, M. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.* **31**, 1–8 (2012).
27. Bouaziz, S., Martin, S., Liu, T., Kavan, L. & Pauly, M. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.* **33**, 1–11 (2014).
28. Solenthaler, B. & Gross, M. Two-scale particle simulation. *ACM Trans. Graph.* (2011).
29. Muller, M., Charypar, D. & Gross, M. Particle-based fluid simulation for interactive applications. In *SCA*, 154–159 (2003).
30. Kim, K.-H., Lee, J., Kim, C.-H. & Kim, J.-H. Visual simulation of turbulent foams by incorporating the angular momentum of foam particles into the projective framework. *Appl. Sci.* **12**, 133 (2021).
31. Wang, H. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.* **34**, 1–9 (2015).
32. Akinci, N., Cornelis, J., Akinci, G. & Teschner, M. Coupling elastic solids with smoothed particle hydrodynamics fluids. *Comput. Animation Virtual Worlds* **24**, 195–203 (2013).
33. Bridson, R., Fedkiw, R. & Anderson, J. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH*, 594–603 (2002).

Author contributions

J.H.K. conceived the study and developed the core methodology. J.H.K. and J.L. implemented the system and conducted the experiments. J.H.K. analyzed the results and wrote the manuscript. All authors reviewed the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to J.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.