

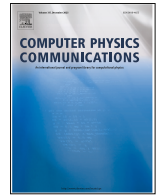


Title	GPU-accelerated micropolar δ^+ -SPH framework with enhanced particle shifting for complex fluid dynamics
Author(s)	Chen, Yixuan; Shao, Qiran; Sugiyama, Yoshie
Citation	Computer Physics Communications. 2026, 322, p. 110081
Version Type	VoR
URL	https://hdl.handle.net/11094/104610
rights	This article is licensed under a Creative Commons Attribution 4.0 International License.
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka



Computational Physics

GPU-accelerated micropolar δ^+ -SPH framework with enhanced particle shifting for complex fluid dynamics

Yixuan Chen , Qiran Shao , Yoshie Sugiyama *

Department of Pure and Applied Mathematics, Osaka University, 2-1 Yamadaoka, Suita, Osaka, 565-0871, Japan

ARTICLE INFO

Editor: Prof. Andrew Hazel

PACS:

07.05.Tp

02.70.-c

07.05.Kf

47.11.-j

Keywords:

Smoothed particle hydrodynamics

Weakly-compressible SPH

 δ^+ -SPH

GPU Computing

Cross-platform

Micropolar fluids

Computational fluid dynamics

OpenGL compute shaders

ABSTRACT

Smoothed Particle Hydrodynamics (SPH) faces persistent challenges in computational efficiency and numerical accuracy. This paper presents a GPU-accelerated framework integrating micropolar δ^+ -SPH with enhanced stabilization techniques. The implementation features four key components: cross-platform execution on both NVIDIA and AMD GPUs via OpenGL compute shaders; an in-out buffer system enabling fully parallel inter-voxel particle migration without global synchronization; an enhanced particle shifting technique maintaining uniform distributions without explicit free-surface detection; and a micropolar formulation preserving vortical structures through bidirectional coupling between velocity and micro-rotation fields. These design choices yield substantial improvements: (a) benchmarks demonstrate 17–18% faster execution than DualSPHysics v5.4 and 97× speedup over server-class CPU implementations; (b) lid-driven cavity simulations achieve L_2 errors below 1.3% for Reynolds numbers 100–10,000, while NACA0015 airfoil simulations at $Re = 180,000$ show lift coefficient agreement within 7% of reference data; (c) the micropolar formulation reduces aerodynamic coefficient fluctuations by 20–22% at stall conditions compared to classical Navier–Stokes; and (d) the zero-copy architecture eliminates I/O bottlenecks, enabling real-time visualization and parameter adjustment. This framework establishes SPH as a practical tool for real-time fluid dynamics simulation, bridging the gap between academic research and industrial applications.

Program summary

Program Title: GPU-Micropolar-DeltaPlus-SPH*Developer(s):* Chen Yixuan*Licensing provisions:* MIT License*Programming language:* Python, GLSL (OpenGL Shading Language)*Computer:* Desktop workstation or HPC system with OpenGL 4.5 + support*Operating system:* Linux, Windows 10/11*RAM:* Minimum 8 GB, recommended 32 GB*GPU Memory:* Minimum 4 GB, recommended 12 GB or more*Keywords:* SPH, GPU computing, Micropolar fluids, δ^+ -SPH, CFD*Classification:* 02.70.-c (Computational techniques; simulations), 07.05.Tp (Computer modeling and simulation), 02.70.Ns (Molecular dynamics and particle methods), 47.11.Kb (Spectral and particle methods)*External routines:* OpenGL 4.5 +, GLFW 3.3 +*Nature of problem:* Simulation of incompressible viscous flows at high Reynolds numbers with complex vortical structures, free surfaces, and moving boundaries*Solution method:* GPU-accelerated SPH with micropolar formulation, δ^+ -SPH stabilization, and enhanced particle shifting*Unusual features:* Zero-copy visualization pipeline, cross-platform GPU compatibility via OpenGL, adaptive particle shifting without free-surface detection*Running time:* Real-time (300 + FPS) for 10^6 particles on NVIDIA RTX 5090

* Corresponding authors.

E-mail addresses: chen.yixuan@ist.osaka-u.ac.jp (Y. Chen), shao.qiran@ist.osaka-u.ac.jp (Q. Shao), sugiyama.ist@osaka-u.ac.jp (Y. Sugiyama).<https://doi.org/10.1016/j.cpc.2026.110081>

Received 24 September 2025; Received in revised form 7 January 2026; Accepted 10 February 2026

Available online 14 February 2026

0010-4655/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Smoothed Particle Hydrodynamics (SPH), introduced independently by Lucy [1] and Gingold and Monaghan [2] in 1977, has evolved from its astrophysical origins into a versatile mesh-free method for computational fluid dynamics. As a fully Lagrangian particle approach, SPH offers unique advantages in simulating flows with free surfaces, large deformations, and moving boundaries, which represent scenarios where traditional grid-based methods face significant challenges [3,4]. Today, SPH has been widely applied in diverse applications ranging from coastal engineering and automotive crash analysis to geophysics and biomedical flows.

Nevertheless, SPH has not yet become a mainstream tool for industrial-scale CFD, primarily due to two persistent challenges: computational cost and numerical fidelity. On the computational side, each particle typically interacts with 50 to 100 neighbors at every timestep, creating workloads that are both computationally intensive and strongly memory-bound [5,6]. CPU-based implementations, even with advanced parallelization strategies, often require hours of computational time to simulate mere seconds of physical phenomena [7,8]. On the numerical side, standard SPH formulations suffer from artificial dissipation, tensile instability, and kernel truncation errors, which significantly degrade solution accuracy [9–11]. Addressing these limitations typically requires advanced techniques such as δ^+ -SPH, micropolar models, or kernel gradient corrections. However, most existing implementations support only a single formulation, thereby limiting systematic method comparisons and the development of hybrid approaches.

Recent advances in GPU computing have demonstrated substantial potential for addressing SPH's computational bottlenecks. Modern GPUs provide memory bandwidth of 1.5 to 3.0 TB/s through high-bandwidth memory (HBM) architectures, representing an order-of-magnitude improvement over the 100 to 200 GB/s typical of CPU-based DDR5 systems. Furthermore, their thousands of processing cores enable massive parallelism to minimize latency [5,6]. This hardware advantage proves particularly critical for SPH workloads, which are characterized by irregular memory access patterns that render CPU cache hierarchies largely ineffective. Consequently, GPU-based SPH implementations have achieved performance improvements ranging from two to three orders of magnitude relative to optimized CPU solvers [12,13], establishing hardware acceleration as an essential requirement for simulations involving millions of particles.

Significant progress has been made in GPU-accelerated SPH over the past decade. Domínguez et al. [14] developed multi-GPU domain decomposition with inter-GPU communication strategies, while Xia and Liang [15] demonstrated GPU-parallel SPH for shallow-water simulations. More recently, Morikawa et al. [16] extended explicit incompressible SPH to multi-GPU clusters for large-scale simulations, and Cai et al. [17,18] addressed CUDA-level optimizations including memory access patterns, thread scheduling, and GPU memory bottlenecks. Mao et al. [19] introduced mixed-precision FP16/FP32 computation with cell-based relative coordinates to further improve efficiency. Open-source frameworks such as DualSPHysics [12,20] and SPLisH-SPlasH [21] have made GPU-accelerated SPH widely accessible, providing robust implementations for free-surface flows and multiphysics problems.

Despite these advances, existing GPU-SPH implementations remain highly fragmented. Numerous frameworks rely on vendor-specific technologies, such as CUDA for NVIDIA hardware or ROCm for AMD systems, which significantly hinders portability across heterogeneous computing platforms. Furthermore, many implementations maintain a decoupled architecture between computation and visualization modules, necessitating the transfer of gigabytes of particle data to disk prior to rendering. This architectural separation creates severe I/O bottlenecks that fundamentally prevent real-time analysis [22]. Such disconnection proves particularly problematic when simulating transient flow phenomena, including vortex shedding and shock wave propagation, where

critical flow features may emerge and dissipate within the temporal intervals between consecutive output frames.

Beyond computational and I/O challenges, numerical accuracy and stability constitute fundamental obstacles for SPH implementations. Classical SPH formulations inherently dampen vorticity and suppress turbulent structures, particularly at coarse spatial resolutions [9]. Tensile instability manifests under negative pressure conditions, inducing unphysical particle clustering that ultimately leads to simulation failure [10]. Boundary treatment methodologies, including ghost particle approaches [23] and semi-analytical wall formulations [24], introduce additional algorithmic complexity to the computational framework. Consequently, robust SPH simulations necessitate the simultaneous integration of multiple advanced techniques, including δ -SPH with its diffusion coefficient selection, micropolar models requiring transfer and micro-inertia parameters, kernel gradient corrections demanding matrix inversions, and particle shifting schemes with their CFL-dependent coefficients. Each technique introduces distinct computational overhead and parameter sensitivities that must be carefully balanced.

To the best of our knowledge, no existing GPU-accelerated SPH framework simultaneously provides cross-platform portability, efficient spatial data structures optimized for GPU architectures, integrated support for multiple advanced SPH formulations including micropolar fluids, and a unified simulation-visualization pipeline capable of real-time performance. This work presents a comprehensive GPU-accelerated SPH framework that directly addresses these limitations through four key innovations: (i) cross-platform portability through OpenGL compute shaders, ensuring seamless deployment across NVIDIA, AMD, and Intel GPU architectures without vendor-specific code branches or performance degradation; (ii) a voxel-based spatial data structure with an in-out buffer system that enables fully parallel inter-voxel particle migration without global synchronization, achieving constant-time average-case neighbor searches and cache-coherent memory access patterns; (iii) unified support for multiple SPH formulations within a single computational pipeline, including classical SPH, δ -SPH, δ^+ -SPH, and micropolar SPH, facilitating systematic comparisons and the development of hybrid schemes; and (iv) zero-copy architecture maintaining particle data exclusively in GPU memory with direct coupling between computation and visualization pipelines, eliminating CPU-GPU data transfers and enabling interactive exploration of transient flow phenomena.

Performance benchmarks demonstrate substantial improvements in both computational efficiency and numerical accuracy. Direct comparison against DualSPHysics v5.4 reveals 17–18% reduction in wall-clock time for equivalent test cases on the same GPU hardware, while speedups of 97 $\times$ over server-class CPU implementations (Intel Xeon Gold 6226R) approach two orders of magnitude. Cross-platform validation on both NVIDIA RTX 5090 and AMD RX 7900 XTX hardware confirms true vendor independence, achieving speedups of 37 $\times$ and 41 $\times$ over consumer CPU baselines respectively. Comprehensive validation studies confirm numerical accuracy across diverse flow regimes: lid-driven cavity simulations achieve relative L_2 errors below 1.3% compared to high-resolution finite difference benchmarks across Reynolds numbers from 100 to 10,000, with spatial convergence rates of 1.3–1.6 consistent with theoretical expectations. External flow validations demonstrate mean drag coefficient differences of 6.7% for elliptical cylinders compared to viscous vortex domains method reference data across Reynolds numbers from 10^4 to 10^6 . NACA0015 airfoil simulations at $Re = 180,000$ show lift coefficient agreement within 7% of reference solutions, with the micropolar formulation reducing temporal fluctuations in aerodynamic coefficients by 20–22% at stall conditions compared to classical Navier–Stokes through its bidirectional coupling between velocity and micro-rotation fields.

Through the integration of high computational efficiency with advanced numerical formulations and real-time visualization capabilities, this framework establishes SPH as a viable tool for industrial-scale computational fluid dynamics applications, effectively bridging the gap between academic research and practical engineering requirements.

The remainder of this paper is organized as follows. Section 2 presents the mathematical formulation, including the micropolar fluid model and enhanced particle shifting techniques. Section 3 describes the GPU implementation strategy, focusing on voxel-based spatial data structures and performance characterization. Section 4 validates the framework through comprehensive benchmark test cases. Section 5 examines the implications of our findings, addresses current limitations, and identifies future research directions. Finally, Section 6 summarizes the key contributions and their significance for computational fluid dynamics.

2. Mathematical formulation

This section presents a comprehensive mathematical framework for our GPU-accelerated SPH implementation. We begin with the fundamental SPH equations and their discretization, followed by the micropolar fluid formulation to address numerical diffusion. We then introduce our enhanced δ^+ -SPH scheme with novel particle-shifting mechanisms. The section concludes with the complete discretized system that integrates all these advances into a unified computational framework.

2.1. Fundamental SPH equations

2.1.1. Kernel approximation theory

The foundation of SPH lies in the integral representation of field functions via the kernel approximation method. For any continuous field function $f(\mathbf{r})$ in the domain $\Omega \subset \mathbb{R}^d$ with $d = 2, 3$, the exact representation through the Dirac delta function reads:

$$f(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}') \delta(\mathbf{r} - \mathbf{r}') d\mathbf{r}'. \quad (1)$$

In SPH, this delta distribution is replaced by a smoothing kernel function W with compact support, yielding the mollified approximation [25]:

$$\langle f \rangle(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}') W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}', \quad (2)$$

where h is the smoothing length that determines the characteristic length of the kernel support. The angle bracket notation $\langle \cdot \rangle$ denotes the SPH approximation of the enclosed quantity.

The kernel function must satisfy several standard properties including normalization ($\int W d\mathbf{r}' = 1$), compact support, positivity, symmetry, and sufficient smoothness (C^k with $k \geq 2$); see [25–27] for details.

A crucial property for the development of SPH differential operators is the radial symmetry of the kernel function: W depends only on the distance $\tilde{r} := \|\mathbf{r} - \mathbf{r}'\|$. Consequently, its gradient takes the form [25]:

$$\nabla W(\mathbf{r} - \mathbf{r}', h) = -(\mathbf{r} - \mathbf{r}') F(\tilde{r}, h), \quad (3)$$

where the scalar function F is defined by:

$$F(\tilde{r}, h) := \frac{\nabla W \cdot (\mathbf{r}' - \mathbf{r})}{\tilde{r}^2}. \quad (4)$$

The function $F(\tilde{r}, h)$ is, like W , strictly positive within the kernel support and vanishes at its boundary. From the symmetry of the kernel function with respect to its arguments,

$$W(\mathbf{r} - \mathbf{r}', h) = W(\mathbf{r}' - \mathbf{r}, h), \quad (5)$$

it follows that:

$$\nabla W(\mathbf{r} - \mathbf{r}', h) = -\nabla' W(\mathbf{r} - \mathbf{r}', h), \quad (6)$$

where ∇' denotes the gradient with respect to the variable $\mathbf{r}'$. This anti-symmetry property is essential for ensuring momentum conservation in SPH discretizations.

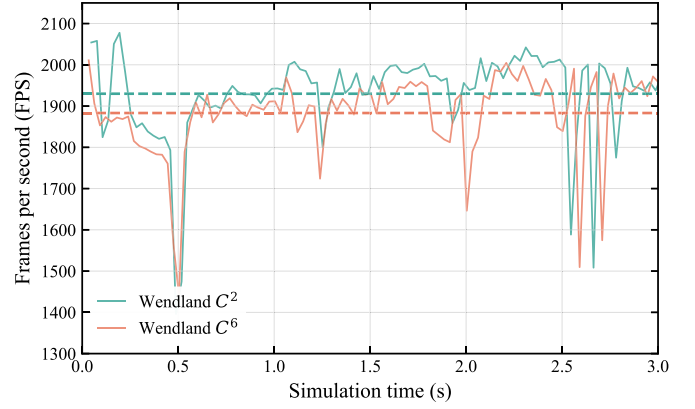


Fig. 1. Performance comparison between Wendland C^2 and C^6 kernels for a dam-break simulation with 23,100 particles. The C^6 kernel achieves a mean FPS of 1883 ± 97 , compared to 1930 ± 109 for the C^2 kernel, representing only a 2.4% performance overhead.

2.1.2. Wendland C^6 kernel function example

Among the various kernel functions available in SPH, the Wendland family of kernels offers excellent stability and accuracy characteristics due to their compact support and high-order continuity properties. We demonstrate the kernel approximation theory using the Wendland C^6 kernel first introduced by Wendland [28], which provides $C^6(\mathbb{R}^d)$ -continuity. Dehnen and Aly [29] demonstrated that higher-order Wendland kernels offer improved convergence rates, and we select Wendland C^6 for its optimal balance between smoothness and computational requirements for our GPU implementation.

While the C^2 Wendland kernel is more commonly adopted in the SPH literature due to its lower computational cost, we deliberately choose the higher-order C^6 variant for two reasons. First, in our GPU-accelerated implementation, the additional arithmetic operations required for evaluating the C^6 kernel result in only a marginal performance overhead of approximately 2–3% compared to the C^2 kernel (reducing mean FPS from ~ 1930 to ~ 1880 , as shown in Fig. 1 for a simple DamBreak experiment), a trade-off we consider acceptable given the accuracy benefits. Second, and more importantly, recent studies have demonstrated that kernel smoothness significantly affects the ability of SPH to capture shear-driven instabilities. Tricco [30] investigated the Kelvin-Helmholtz instability using SPH with B-spline kernels of varying order (cubic through nonic) and found that lower-order kernels introduce substantial errors that degrade solution quality, whereas higher-order kernels (septic and above) yield solutions that are indistinguishable from reference solutions. Since our micropolar formulation specifically targets the accurate resolution of vortical structures and rotational dynamics, employing a sufficiently smooth kernel is essential to ensure that numerical errors from the kernel approximation do not obscure the physical phenomena we aim to capture.

The Wendland C^6 kernel [28,29] is defined as:

$$W(\mathbf{r}, h) = \begin{cases} \frac{\alpha_d}{h^d} (1 - q)^8 (32q^3 + 25q^2 + 8q + 1) & \text{for } 0 \leq q \leq 1, \\ 0 & \text{for } q > 1, \end{cases} \quad (7)$$

where $\mathbf{r} \in \Omega \subset \mathbb{R}^d$, $q = \frac{|\mathbf{r}|}{h}$ is the normalized distance and h is the smoothing length that defines the compact support radius. The normalization constant α_d ensures that $\int_{\mathbb{R}^d} W(\mathbf{r}, h) d\mathbf{r} = 1$ for all $h > 0$, satisfying the partition of unity requirement fundamental to SPH approximation theory:

$$\alpha_d = \begin{cases} \frac{78}{7\pi} & \text{for } d = 2, \\ \frac{1365}{64\pi} & \text{for } d = 3. \end{cases} \quad (8)$$

For practical implementation, the smoothing length h is typically chosen as $h = \kappa \Delta x$, where Δx denotes the initial particle spacing and

κ is a scaling factor, typically 1.2 to 2.0, chosen to maintain approximately 40–60 neighboring particles within the kernel support in 3D applications [31], thereby optimizing the balance between computational efficiency and numerical accuracy.

2.1.3. Particle approximation

In the discrete-particle approximation, the continuous domain is represented by a finite set of particles, each carrying physical properties such as mass, density, and velocity. The fundamental principle of SPH transforms the continuous integral representation into a discrete summation over neighboring particles, establishing a mesh-free computational framework.

The integral approximation in Eq. (2) is discretized by replacing the continuous volume element $d\mathbf{r}'$ with discrete particle volumes V_j . The domain Ω is partitioned into elementary volumes, with the j th volume centered at position $\mathbf{r}_j$ and having volume $V_j = m_j/\rho_j$, where m_j and ρ_j denote the mass and density of particle j , respectively. The discretized SPH approximation then reads [25]:

$$\langle f \rangle_i = \sum_{j \in \mathcal{N}_i} f_j W_{ij} V_j, \quad (9)$$

where the compact notation $W_{ij} := W(\tilde{r}_{ij}, h)$ denotes the kernel evaluated at the inter-particle distance $\tilde{r}_{ij} := \|\mathbf{r}_i - \mathbf{r}_j\|$. The index set $\mathcal{N}_i$ comprises all neighboring particles within the kernel support, i.e., those satisfying $\tilde{r}_{ij} \leq h$, where h is the support radius. The kernel symmetry $W_{ij} = W_{ji}$ ensures that interaction strengths between particle pairs are identical regardless of the direction of evaluation.

This formulation provides a flexible framework for approximating field variables and their derivatives without requiring explicit connectivity information between particles. The approximation naturally adapts to the local particle distribution, with accuracy primarily dependent on the regularity of the particle arrangement, the smoothing length selection, and the kernel function choice. The method effectively handles spatial derivatives, along with other aspects, by shifting differential operations from unknown field functions to analytically known kernel functions, as explored in the following sections.

2.1.4. Differential operators approximation in SPH

The computation of spatial derivatives is fundamental to solving the governing equations of fluid dynamics. In SPH, differential operators are approximated by transferring the derivatives from the field function to the kernel function, which is known analytically. This transfer is justified by the properties of the convolution integral and allows us to compute derivatives without requiring connectivity information between particles. However, this transfer introduces several challenges that must be addressed for accurate simulations.

Central to all differential operator approximations is the discrete form of the kernel gradient. Exploiting the radial symmetry property (3), the kernel gradient can be expressed as:

$$\nabla_i W_{ij} = -\mathbf{r}_{ij} F_{ij}, \quad F_{ij} := F(\tilde{r}_{ij}, h), \quad (10)$$

where $\mathbf{r}_{ij} := \mathbf{r}_i - \mathbf{r}_j$ denotes the position vector from particle j to particle i . The antisymmetry $\mathbf{r}_{ij} = -\mathbf{r}_{ji}$ combined with the symmetry $F_{ij} = F_{ji}$ ensures that $\nabla_i W_{ij} = -\nabla_j W_{ij}$, a property fundamental to momentum conservation. This decomposition provides a unified framework for all differential operators: while gradient and divergence approximations employ the full kernel gradient $\nabla_i W_{ij} = -\mathbf{r}_{ij} F_{ij}$, the Laplacian operator (19) requires only the scalar function F_{ij} , yielding a particularly compact formulation [25].

Gradient Approximation

The SPH gradient approximation is obtained by transferring the gradient operator from the field function to the kernel. When the kernel support is fully contained within the fluid domain Ω , this yields:

$$\langle \nabla f \rangle(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}') \nabla W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}'. \quad (11)$$

When the kernel support intersects a domain boundary $\partial\Omega$, such as a free surface, the complete expression includes a boundary integral term [25]:

$$\langle \nabla f \rangle(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}') \nabla W d\mathbf{r}' + \int_{\partial\Omega} f(\mathbf{r}') W \mathbf{n}' dS', \quad (12)$$

where $\mathbf{n}'$ is the outward normal to $\partial\Omega$. The surface integral vanishes for all points $\mathbf{r}$ inside Ω whose kernel supports do not intersect the domain boundary; for particles near free surfaces, this kernel truncation introduces consistency errors that are addressed by the kernel gradient correction discussed below.

Discretizing using the particle approximation and applying (10) yields:

$$\langle \nabla f_i \rangle = \sum_{j \in \mathcal{N}_i} f_j \nabla_i W_{ij} V_j = - \sum_{j \in \mathcal{N}_i} f_j \mathbf{r}_{ij} F_{ij} V_j, \quad (13)$$

where $V_j = m_j/\rho_j$ is the particle volume. This standard formulation exhibits zero-order inconsistency near boundaries and lacks the antisymmetry required for momentum conservation [31,32]. To address these limitations, two improved gradient formulations are commonly employed.

The symmetric formulation reads:

$$\langle \nabla f_i \rangle = \sum_{j \in \mathcal{N}_i} (f_i + f_j) \nabla_i W_{ij} V_j = - \sum_{j \in \mathcal{N}_i} (f_i + f_j) \mathbf{r}_{ij} F_{ij} V_j, \quad (14)$$

where the second equality follows from (10). This form ensures exact conservation of linear and angular momentum through the antisymmetric structure of the pairwise interactions, making it particularly suitable for pressure gradient calculations [31].

The difference formulation is given by:

$$\langle \nabla f_i \rangle = \sum_{j \in \mathcal{N}_i} (f_j - f_i) \nabla_i W_{ij} V_j = - \sum_{j \in \mathcal{N}_i} (f_j - f_i) \mathbf{r}_{ij} F_{ij} V_j. \quad (15)$$

This formulation provides improved accuracy near free surfaces and boundaries by using relative field values, naturally vanishing for constant fields and reducing numerical artifacts in regions with particle deficiency [25].

Divergence Approximation

The divergence of a vector field $\mathbf{A}$ follows analogously from the kernel approximation. The symmetric formulation reads:

$$\langle \nabla \cdot \mathbf{A}_i \rangle = \sum_{j \in \mathcal{N}_i} (\mathbf{A}_i + \mathbf{A}_j) \cdot \nabla_i W_{ij} V_j = - \sum_{j \in \mathcal{N}_i} [(\mathbf{A}_i + \mathbf{A}_j) \cdot \mathbf{r}_{ij}] F_{ij} V_j, \quad (16)$$

which preserves momentum conservation through the antisymmetric pairwise structure [31]. The difference formulation is given by:

$$\langle \nabla \cdot \mathbf{A}_i \rangle = \sum_{j \in \mathcal{N}_i} (\mathbf{A}_j - \mathbf{A}_i) \cdot \nabla_i W_{ij} V_j = - \sum_{j \in \mathcal{N}_i} [(\mathbf{A}_j - \mathbf{A}_i) \cdot \mathbf{r}_{ij}] F_{ij} V_j. \quad (17)$$

This formulation ensures that the divergence of a uniform field vanishes exactly and provides improved accuracy near free surfaces [25]. In our implementation, we employ the difference formulation combined with renormalization techniques.

Curl Approximation

For vorticity calculations in our micropolar formulation, the curl operator employs the difference formulation:

$$\langle \nabla \times \mathbf{A}_i \rangle = \sum_{j \in \mathcal{N}_i} (\mathbf{A}_j - \mathbf{A}_i) \times \nabla_i W_{ij} V_j = \sum_{j \in \mathcal{N}_i} (\mathbf{A}_i - \mathbf{A}_j) \times \mathbf{r}_{ij} F_{ij} V_j. \quad (18)$$

This formulation properly yields zero curl for irrotational flows and maintains accuracy near free surfaces.

Laplacian Approximation

The mollified Laplacian operator takes a particularly compact form using only the scalar function F_{ij} [25]:

$$\langle \nabla^2 f_i \rangle = 2 \sum_{j \in \mathcal{N}_i} (f_j - f_i) F_{ij} V_j. \quad (19)$$

This formulation provides $\mathcal{O}(h^2)$ consistency for interior particles. Note that for vector field diffusion (e.g., viscous forces), a variationally-derived formulation is required to ensure proper conservation properties, as discussed in the discretization of the momentum equation.

Kernel Gradient Correction

Despite improvements from symmetric and difference formulations, SPH gradient calculations lose accuracy near boundaries and in regions with irregular particle distributions. To address this limitation, we implement kernel gradient correction (KGC) [25,32], which ensures exact gradient computation for linear functions. The SPH gradient operator must satisfy the consistency condition:

$$\sum_{j \in \mathcal{N}_i} \mathbf{r}_{ij} \otimes \mathbf{r}_{ij} F_{ij} V_j = \mathbf{I}, \quad (20)$$

where $\mathbf{I}$ is the identity matrix. This condition is rarely satisfied naturally, especially near boundaries. We therefore compute a symmetric correction matrix $\mathbf{L}_i$ [25,32]:

$$\mathbf{L}_i = \left[\sum_{j \in \mathcal{N}_i} \mathbf{r}_{ij} \otimes \mathbf{r}_{ij} F_{ij} V_j \right]^{-1}. \quad (21)$$

The corrected gradient, combining KGC with the difference formulation, reads:

$$\langle \nabla f_i \rangle^L = \sum_{j \in \mathcal{N}_i} (f_j - f_i) \mathbf{L}_i \cdot \nabla_i W_{ij} V_j. \quad (22)$$

This formulation ensures both first-order consistency (exact gradients for linear functions) and zero-order consistency (vanishing gradients for constant fields), even where the kernel support is truncated. The matrix inversion (2×2 in 2D, 3×3 in 3D) incurs modest overhead, and $\mathbf{L}_i$ can be reused across all differential operators within the same timestep.

This correction is essential for high-fidelity simulations, particularly for complex flows with strong gradients.

2.1.5. Governing equations

Having established the SPH framework for approximating field quantities and their derivatives, we now apply this methodology to the fundamental equations governing fluid dynamics. The Lagrangian nature of SPH, where computational particles move with the fluid, provides a natural setting for expressing the conservation laws. In this framework, the Navier-Stokes equations take a form that directly tracks the evolution of fluid properties following the material particles.

The Continuity Equation

In the Lagrangian formulation, conservation of mass is expressed through the continuity equation in its material derivative form:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}, \quad (23)$$

where ρ is the fluid density, $\mathbf{v}$ is the velocity vector, and $\frac{D}{Dt}$ denotes the material derivative following the fluid particles. This form differs from the Eulerian representation by eliminating the convective term, as the derivative inherently follows the particle motion. In Lagrangian coordinates, the material derivative naturally accounts for the advection of properties with the flow, simplifying the mathematical structure while maintaining physical completeness.

The Momentum Equation

The conservation of momentum in Lagrangian form is governed by:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{g}, \quad (24)$$

where p is the pressure, ν is the kinematic viscosity, and $\mathbf{g}$ represents the external body forces per unit mass. The pressure gradient term $-\frac{1}{\rho} \nabla p$ drives the flow from regions of high pressure to low pressure, providing the primary mechanism for momentum transfer in inviscid flows. The viscous diffusion term $\nu \nabla^2 \mathbf{v}$ accounts for momentum diffusion due to

molecular viscosity, becoming particularly important in boundary layers and regions of strong velocity gradients. The external force term $\mathbf{g}$ incorporates body forces per unit mass, typically gravity, though it may also include other volumetric forces such as electromagnetic or centrifugal effects depending on the application.

SPH Discretization of Governing Equations

The discretization of the governing equations transforms the continuous differential operators into summations over neighboring particles.

Density Computation

Two independent approaches exist for computing the density field. The first directly evaluates density from the particle distribution:

$$\langle \rho_i \rangle = \sum_{j \in \mathcal{N}_i} m_j W_{ij}. \quad (25)$$

The second approach discretizes the continuity Eq. (23) using the velocity divergence approximation. Applying the difference formulation (17) yields [25]:

$$\frac{d\rho_i}{dt} = -\rho_i \sum_{j \in \mathcal{N}_i} (\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}) F_{ij} V_j, \quad (26)$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ is the relative velocity. This formulation, corresponding to Scheme S₅ in Le Touzé and Colagrossi [25], is consistent even when the kernel support intersects the free surface, and ensures Galilean invariance through the use of relative velocities.

Momentum Equation Discretization

The SPH discretization of the momentum equation requires careful treatment of the pressure gradient and viscous terms to ensure momentum conservation. Following Scheme S₅ [25], the pressure gradient term is discretized as:

$$\mathbf{f}_i^p = \frac{1}{\rho_i} \sum_{j \in \mathcal{N}_i} (p_i + p_j) \mathbf{r}_{ij} F_{ij} V_j, \quad (27)$$

where the symmetric form $(p_i + p_j)$ ensures exact conservation of linear and angular momentum through its antisymmetric structure with respect to particle pairs, satisfying Newton's third law at the discrete level.

The viscous term is discretized following the variational derivation in [25], which ensures preservation of angular momentum and zero viscous force for rigid body motion. Direct application of a simple mollified Laplacian to the velocity field does not satisfy these requirements [25]. The variationally-derived formulation reads:

$$\mathbf{f}_i^v = -\mathcal{K} \nu \sum_{j \in \mathcal{N}_i} \pi_{ij} \mathbf{r}_{ij} F_{ij} V_j, \quad (28)$$

where $\mathcal{K} = 2(d+2)$ is a dimensional consistency factor with d being the number of spatial dimensions and the quantity π_{ij} is defined as:

$$\pi_{ij} := \frac{\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2}, \quad (29)$$

with $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ being the relative velocity. This formulation is derived from a dissipation function using a variational principle, guaranteeing thermodynamic consistency [25].

Complete Discrete System

Combining these discretizations, the complete SPH system corresponding to Scheme S₅ [25] becomes:

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \mathbf{v}_i, \\ \frac{d\rho_i}{dt} &= -\rho_i \sum_{j \in \mathcal{N}_i} (\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}) F_{ij} V_j, \quad V_i(t) = \frac{m_i}{\rho_i(t)}, \\ \frac{d\mathbf{v}_i}{dt} &= \frac{1}{\rho_i} \sum_{j \in \mathcal{N}_i} (p_i + p_j) \mathbf{r}_{ij} F_{ij} V_j - \mathcal{K} \nu \sum_{j \in \mathcal{N}_i} \pi_{ij} \mathbf{r}_{ij} F_{ij} V_j + \mathbf{g}. \end{aligned} \quad (30)$$

The system is closed by relating pressure to density through an equation of state. For weakly compressible SPH simulations, the Tait equation [33] is commonly adopted:

$$p = \frac{\rho_0 c_0^2}{\gamma} \left[\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right], \quad (31)$$

where ρ_0 is the reference density, $\gamma = 7$ for water-like fluids, and c_0 is the artificial sound speed. The sound speed is typically chosen as $c_0 \geq 10 \max |\mathbf{v}|$ to maintain the Mach number below 0.1, ensuring that density variations remain within approximately 1% while avoiding the computational cost of solving a Poisson equation for pressure.

The discretization presented above follows the framework developed in [25], adapted for weakly compressible isothermal flows. The key features of this formulation include: (i) the continuity-based density evolution (26), which maintains consistency at free surfaces where kernel truncation occurs; (ii) the symmetric pressure gradient, which ensures momentum conservation; and (iii) the variationally-derived viscous force, which preserves angular momentum and vanishes for rigid body motion. These properties provide a robust foundation for the enhanced schemes presented in the following sections.

Beyond Classical SPH

The formulations presented above represent the standard SPH discretization of the Navier-Stokes equations. However, classical SPH suffers from numerical diffusion that can suppress important flow features, particularly in turbulent regimes. In the following sections, we will extend these equations to address these limitations through several advanced techniques. The micropolar fluid formulation will be introduced to capture rotational dynamics and reduce numerical diffusion, providing a more accurate representation of vortical structures. We will implement the δ^+ -SPH scheme with density diffusion terms for pressure stabilization, effectively eliminating spurious pressure oscillations that plague standard formulations. Additionally, enhanced particle-shifting techniques will be employed to maintain regular particle distributions throughout the simulation, preventing particle clustering and void formation that can degrade solution accuracy. These extensions work synergistically to overcome the fundamental limitations of classical SPH while preserving its advantageous conservation properties and computational efficiency.

2.2. Micropolar SPH formulation

To address the numerical diffusion problems inherent in classical SPH, we incorporate the micropolar fluid model. This approach, pioneered by Eringen [34] in his foundational theory of micropolar fluids, extends the classical continuum mechanics by considering the microstructure of fluid particles. The micropolar formulation has demonstrated significant advantages in capturing complex flow phenomena, particularly in flows with dominant vortical structures, blood flow modeling where the micropolar theory accounts for the microstructure effects of red blood cells and their rotational motion [35], and flows through porous media [36].

In the context of SPH, Bender et al. [9] presented a comprehensive micropolar SPH formulation for computer graphics applications, demonstrating that the micro-rotation field effectively serves as both a source and sink for vorticity. The micropolar formulation introduces an independent micro-rotation field that provides additional rotational degrees of freedom, effectively serving as a vorticity reservoir that counteracts the excessive numerical dissipation inherent in standard SPH discretizations. This mechanism proves particularly advantageous at moderate spatial resolutions, where classical SPH formulations exhibit rapid diffusion of vortical structures; the bidirectional coupling between the velocity and micro-rotation fields enables sustained preservation of coherent vortices that would otherwise decay prematurely. Souto-Iglesias et al. [37] subsequently developed a rigorous bottom-up derivation from the Lagrangian formalism, establishing the most general isotropic mi-

cropolar SPH model with guaranteed conservation of linear and angular momentum.

2.2.1. Governing equations

The micropolar fluid equations extend the classical Navier-Stokes momentum equation and introduce an additional transport equation for micro-rotation.

The enhanced momentum equation

$$\frac{D\mathbf{v}}{Dt} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{v} + (\nu + \nu_t) \nabla \times \mathbf{w} + \mathbf{g}. \quad (32)$$

The micro-rotation transport equation

$$\Theta \frac{D\mathbf{w}}{Dt} = \xi \nabla^2 \mathbf{w} + (\nu + \nu_t)(\nabla \times \mathbf{v} - 2\mathbf{w}) + \mathbf{f}, \quad (33)$$

where $\mathbf{w}$ is the micro-rotation field, ν_t is the transfer coefficient controlling the coupling between linear and angular velocities, Θ is the micro-inertia coefficient, ξ is the angular viscosity coefficient, and $\mathbf{f}$ represents external torques acting on the micro-rotation field. In this work, we set the micro-inertia coefficient to $\Theta = 2$, a value that has proven effective in previous micropolar SPH implementations [9] and provides stable numerical behavior, though experimental calibration using tracer particles remains an alternative approach [38].

The key advantage of the micropolar formulation lies in the additional transport equation for micro-rotation, which provides a mechanism for vorticity preservation and generation. The coupling term $(\nu + \nu_t) \nabla \times \mathbf{w}$ in the momentum equation introduces rotational effects that help maintain turbulent structures, while the source term $(\nu + \nu_t)(\nabla \times \mathbf{v} - 2\mathbf{w})$ in the micro-rotation equation ensures proper coupling between the velocity and micro-rotation fields.

SPH Discretization of Micropolar Equations

Enhanced Momentum Equation Discretization

The enhanced momentum equation is discretized using the established SPH formulations for each term. Following the discretization framework established in Section 2.1, the pressure gradient maintains the symmetric formulation to ensure momentum conservation, while the viscous and micro-rotation coupling terms employ the specialized formulations developed for the respective operators.

The discretized enhanced momentum equation reads:

$$\begin{aligned} \frac{d\mathbf{v}_i}{dt} = & \frac{1}{\rho_i} \sum_{j \in \mathcal{N}_i} (p_i + p_j) \mathbf{r}_{ij} F_{ij} V_j - \mathcal{K} \nu \sum_{j \in \mathcal{N}_i} \pi_{ij} \mathbf{r}_{ij} F_{ij} V_j \\ & + (\nu + \nu_t) \sum_{j \in \mathcal{N}_i} (\mathbf{w}_i - \mathbf{w}_j) \times \mathbf{r}_{ij} F_{ij} V_j + \mathbf{g}, \end{aligned} \quad (34)$$

where $\mathcal{K} = 2(d+2)$ is the dimensional consistency factor and π_{ij} is defined in Eq. (29). The curl of the micro-rotation field $\nabla \times \mathbf{w}$ is discretized using the difference formulation (18).

Micro-rotation Equation Discretization

The micro-rotation transport equation requires careful discretization of both the angular viscosity term and the vorticity coupling.

Following the variational derivation for viscous forces [25,37], the angular diffusion term $\xi \nabla^2 \mathbf{w}$ is discretized analogously to the linear viscous term. We define the quantity for micro-rotation:

$$\Pi_{ij} := \frac{\mathbf{w}_{ij} \cdot \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2}, \quad (35)$$

where $\mathbf{w}_{ij} = \mathbf{w}_i - \mathbf{w}_j$ is the relative micro-rotation. The complete SPH discretization of the micro-rotation equation then becomes:

$$\begin{aligned} \Theta \frac{d\mathbf{w}_i}{dt} = & -\mathcal{K} \xi \sum_{j \in \mathcal{N}_i} \Pi_{ij} \mathbf{r}_{ij} F_{ij} V_j \\ & + (\nu + \nu_t) \left(\sum_{j \in \mathcal{N}_i} (\mathbf{v}_i - \mathbf{v}_j) \times \mathbf{r}_{ij} F_{ij} V_j - 2\mathbf{w}_i \right) + \mathbf{f}_i, \end{aligned} \quad (36)$$

Micropolar Transfer Coefficient Effect on Vortex Preservation in Dam Break Flow

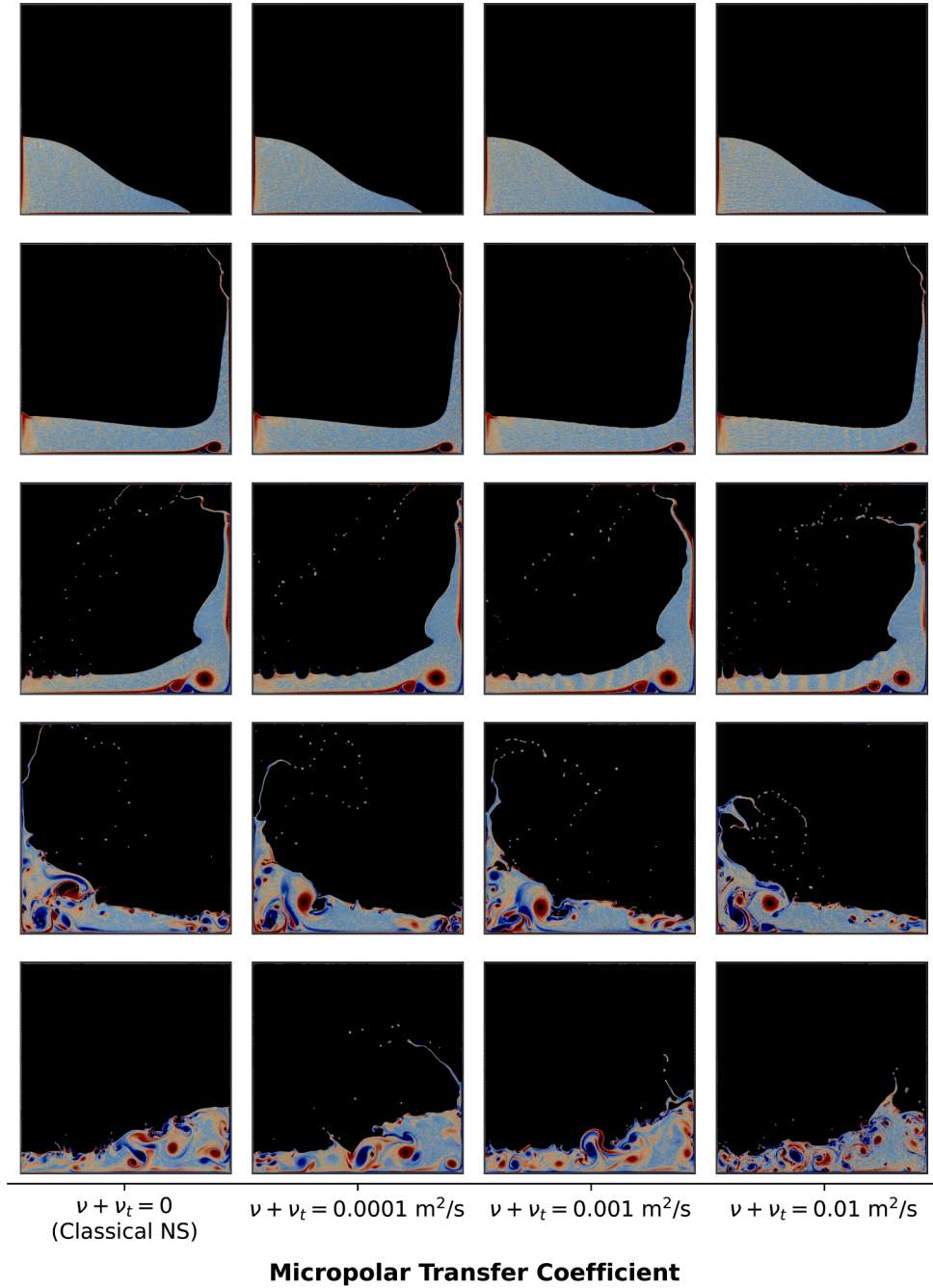


Fig. 2. Dam break simulation demonstrating enhanced vortex preservation with increasing micropolar transfer coefficient. The temporal evolution is shown from top to bottom, with each row capturing the flow field at successive time intervals. From left to right: classical Navier-Stokes formulation ($\nu + \nu_t = 0$) and micropolar formulations with $\nu + \nu_t \in \{10^{-4}, 10^{-3}, 10^{-2}\} \text{ m}^2/\text{s}$. The vorticity magnitude field reveals progressively reduced numerical dissipation of small-scale structures as the transfer coefficient increases. All simulations employ identical initial conditions, boundary conditions, and numerical parameters, with only the micropolar coupling strength varied to isolate its effect on vortex dynamics.

where $\mathcal{K} = 2(d + 2)$ is the same dimensional consistency factor used in the linear viscous term. This formulation maintains consistency with the velocity viscous discretization (28) and preserves angular momentum through its antisymmetric structure [37].

This formulation maintains consistency with the velocity viscous discretization (28) through the parallel structure between π_{ij} for linear viscosity and Π_{ij} for angular viscosity, ensuring that both diffusion mechanisms are treated consistently within the SPH framework.

Fig. 2 offers an exploratory validation of the effect of the micropolar formulation on vortex dynamics. As the transfer coefficient $\nu + \nu_t$ increases from 0, which is the classical Navier-Stokes, to $0.01 \text{ m}^2/\text{s}$, the cascade of small-scale vortical structures exhibits progressively reduced numerical dissipation. The enhanced preservation of rotational dynamics becomes particularly evident in regions of intense shearing, where the micropolar formulation sustains the energy cascade to smaller scales that would otherwise be rapidly damped by numerical diffusion inherent

in classical SPH. A comprehensive analysis of this vorticity-preserving mechanism is presented in [Section 4](#).

2.3. Enhanced δ^+ -SPH formulation with stability control

The δ^+ -SPH scheme represents a significant advancement over the original δ -SPH formulation by incorporating two critical stability control mechanisms: Tensile Instability Control (TIC) and Particle Shifting Technique (PST). In this work, we present an enhanced implementation that introduces a novel PST modification eliminating the need for free-surface detection while maintaining stability only where particles suffer from overlapping and disorder. The enhanced PST also reduces non-physical particle overlapping at free surfaces. This enhanced formulation demonstrates exceptional versatility by seamlessly integrating with both classical Navier-Stokes and micropolar fluid models within a unified computational framework optimized for modern GPU architectures.

2.3.1. δ -SPH framework

The δ -SPH scheme introduces a diffusive term in the continuity equation to regularize pressure oscillations that plague standard SPH formulations.

Density Computation

Following the framework established in [Section 2.1](#), the δ -SPH continuity equation is expressed as:

$$\frac{d\rho_i}{dt} = -\rho_i \sum_{j \in \mathcal{N}_i} (\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}) F_{ij} V_j + 2\delta h c_0 \sum_{j \in \mathcal{N}_i} \Psi_{ij} F_{ij} V_j, \quad (37)$$

where $\delta \approx 0.1$ is the diffusion coefficient, h is the smoothing length, c_0 is the sound speed, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ is the relative velocity, F_{ij} is the scalar kernel function defined in [Eq. \(10\)](#), and $V_j = m_j/\rho_j$ is the particle volume.

The density diffusion term Ψ_{ij} employs a Rusanov-type flux that combines both the density jump and its gradient information [\[10\]](#):

$$\Psi_{ij} = (\rho_j - \rho_i) - \frac{1}{2} (\langle \nabla \rho_i \rangle^L + \langle \nabla \rho_j \rangle^L) \cdot (\mathbf{r}_j - \mathbf{r}_i), \quad (38)$$

where the superscript $\langle \cdot \rangle^L$ indicates that the density gradient is computed using the kernel gradient correction described in [Eq. \(22\)](#):

$$\langle \nabla \rho_i \rangle^L = \sum_{j \in \mathcal{N}_i} (\rho_j - \rho_i) \mathbf{L}_i \cdot \nabla_i W_{ij} V_j. \quad (39)$$

This sophisticated diffusion mechanism operates through two complementary effects. The first term $(\rho_j - \rho_i)$ in [Eq. \(38\)](#) provides direct density smoothing between neighboring particles, effectively damping high-frequency oscillations. The second term, which involves the gradient reconstruction, introduces a higher-order correction that preserves the physical density gradients while filtering numerical noise. This dual action helps reduce spurious oscillations, while the gradient reconstruction term limits the diffusion of physical density variations.

2.3.2. Tensile instability control

Flows with strong velocity gradients often develop regions of negative pressure that trigger tensile instability, leading to particle clustering and simulation failure. To address this, we implement the Tensile Instability Control method introduced in [\[10\]](#) that selectively switches the pressure gradient formulation in unstable regions.

The pressure gradient term is modified according to the local pressure state:

$$\mathbf{f}_i^p = \begin{cases} \frac{1}{\rho_i} \sum_{j \in \mathcal{N}_i} (p_i + p_j) \mathbf{r}_{ij} F_{ij} V_j & \text{if } p_i \geq 0 \text{ or } i \in S_F, \\ \frac{1}{\rho_i} \sum_{j \in \mathcal{N}_i} (p_j - p_i) \mathbf{r}_{ij} F_{ij} V_j & \text{if } p_i < 0 \text{ and } i \notin S_F, \end{cases} \quad (40)$$

where S_F denotes the set of particles near the free surface that have truncated support domains and insufficient neighboring particles. The

algorithm for identifying such particles is detailed in Marrone et al. [\[39\]](#). This approach effectively prevents tensile instability by switching to the difference formulation [\(15\)](#) only in regions where $p_i < 0$ and away from free surfaces, while maintaining the symmetric formulation [\(14\)](#) elsewhere.

2.3.3. Enhanced particle shifting technique

The enhanced PST we introduce in this work represents a fundamental advancement in maintaining particle regularity without requiring explicit free-surface detection. Unlike conventional approaches that require explicit free-surface detection to prevent particle distribution problems [\[40\]](#), our proposed method achieves optimal particle distribution through an adaptive blending mechanism that naturally responds to local particle arrangement quality.

Disorder Parameter

The disorder parameter χ_{ij} quantifies local particle arrangement irregularities:

$$\chi_{ij} = \begin{cases} \frac{\Delta x - \tilde{r}_{ij}}{\Delta x} & \text{if } \tilde{r}_{ij} < \Delta x, \\ 0 & \text{if } \tilde{r}_{ij} \geq \Delta x, \end{cases} \quad (41)$$

where $\tilde{r}_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ is the inter-particle distance as defined in [Section 2.1](#) and Δx represents the initial particle spacing. This formulation ensures that particles closer than the initial spacing contribute to the disorder correction, with the contribution decreasing linearly as particles approach their ideal separation.

Primary Particle Shift

The primary particle shift vector incorporates both the disorder correction and a high-order kernel correction term:

$$\delta \mathbf{r}_i = \alpha \cdot \text{CFL} \cdot h^2 \sum_{j \in \mathcal{N}_i} \left[W_{\text{sum},i} + \chi_{ij} + \beta \left(\frac{W_{ij}}{W(\Delta x, h)} \right)^n \right] \mathbf{r}_{ij} F_{ij} V_j, \quad (42)$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, F_{ij} is the scalar kernel function defined in [Eq. \(10\)](#), and $V_j = m_j/\rho_j$ is the particle volume.

Here, α represents the primary shifting coefficient that controls the overall displacement magnitude, with typical value $\alpha = 0.1$. The Courant-Friedrichs-Lewy number (CFL) ensures numerical stability, while the smoothing length h defines the influence radius for particle interactions. The kernel summation term

$$W_{\text{sum},i} = \sum_{j \in \mathcal{N}_i} W_{ij} V_j \quad (43)$$

measures the completeness of kernel support sampling, with values near unity indicating full kernel coverage in bulk regions and values below unity revealing particle deficiency at boundaries or free surfaces. The disorder parameter χ_{ij} amplifies the shifting strength at irregular locations. The high-order correction term, controlled by coefficient $\beta = 0.2$ with exponent $n = 4$, enhances the accuracy of the shifting scheme in regions with irregular particle distributions.

Anti-Diffusive Component

To counteract excessive particle separation in regions with low disorder, an anti-diffusive shift component is introduced:

$$\delta \mathbf{r}_i^{\text{anti}} = -\gamma \cdot \text{CFL} \cdot h^2 \sum_{j \in \mathcal{N}_i} \left(\frac{\tilde{r}_{ij}}{\Delta x} \right) \mathbf{r}_{ij} F_{ij} V_j, \quad (44)$$

where γ is the anti-diffusive coefficient, we set $\gamma = 5 \times 10^{-3}$ here for our simulation. This component serves a crucial role in maintaining particle cohesion by introducing a mild attractive force that scales with the inter-particle distance. The anti-diffusive shift counters the effect of the primary shift in well-ordered regions where the primary shift would otherwise cause unnecessary particle movement, effectively preventing excessive particle dispersion while maintaining the overall particle distribution quality. We introduce this anti-diffusive mechanism as a novel

solution to the long-standing problem of particle over-dispersion in conventional PST implementations, particularly pronounced at fluid interfaces. This dual-component approach ensures robust performance across diverse flow conditions, from quiescent regions to highly dynamic interfaces.

Adaptive Blending

The final particle displacement is determined through a kernel-summation based blending mechanism:

$$\delta \mathbf{r}_i^{\text{final}} = f(W_{\text{sum},i}) \delta \mathbf{r}_i + [1 - f(W_{\text{sum},i})] \delta \mathbf{r}_i^{\text{anti}}, \quad (45)$$

where the blending function is:

$$f(W_{\text{sum},i}) = [\min(\max(W_{\text{sum},i}, 0), 1)]^3. \quad (46)$$

The cubic blending function provides a smooth transition between the primary and anti-diffusive components based on the local kernel summation value as shown in Fig. 3. When $W_{\text{sum},i}$ approaches unity, indicating a well-sampled bulk region, the primary shift dominates to maintain particle regularity. Conversely, as $W_{\text{sum},i}$ decreases near free surfaces or in particle-deficient regions, the anti-diffusive component becomes more prominent, countering the "explosive-like" behavior of the primary PST term that would otherwise cause excessive particle dispersion. This mild attractive force helps free-surface particles maintain a normal distribution rather than being scattered outward. This adaptive mechanism eliminates the need for explicit free-surface detection while naturally handling the transition between bulk and surface regions.

Fig. 4 demonstrates the superiority of our enhanced PST near free-surface regions. Conventional PST methods (a) employ a Heaviside step function that takes only binary values (0 or 1) based on free-surface detection, where such abrupt transitions cause irregular particle clustering due to uniform shifting forces. In contrast, the proposed enhanced PST (b) maintains the regular arrangement of the particles through the complementary weighting functions shown in Fig. 3. These functions, where one component increases as the other decreases, provide smooth and continuous transitions based on the kernel summation $W_{\text{sum},i}$. This continuous formulation reduces primary shifting as $W_{\text{sum},i}$ decreases, while the anti-diffusive component prevents particle scattering. This comparison confirms that our approach eliminates the need for computationally expensive free-surface detection while maintaining particle regularity at interfaces through smooth complementary weighting functions rather than discontinuous switching.

Algorithm Summary

The Algorithm 1, whose computational efficiency stems from its single-pass nature over the neighbor list, avoiding the computational overhead of free-surface detection algorithms that typically require multiple neighbor searches or complex geometric calculations. The method's robustness is further enhanced by the CFL-based scaling that automatically adjusts the shifting magnitude based on the simulation timestep, ensuring stability across varying flow conditions and particle resolutions.

Algorithm 1 Enhanced particle shifting technique.

```

for all particles  $i$  do
  Initialize  $\delta \mathbf{r}_i = \mathbf{0}$ ,  $\delta \mathbf{r}_i^{\text{anti}} = \mathbf{0}$ 
  for all neighbors  $j$  within smooth radius do
    Compute  $\chi_{ij} = \max\left(0, \frac{\Delta x - r_{ij}}{\Delta x}\right)$ 
    Update  $\delta \mathbf{r}_i$  using Eq. (42)
    Update  $\delta \mathbf{r}_i^{\text{anti}}$  using Eq. (44)
  end for
  Compute blending factor  $f(W_{\text{sum},i})$  using Eq. (46)
  Apply final shift:  $\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \delta \mathbf{r}_i^{\text{final}}$ 
end for

```

2.4. Complete micropolar δ^+ -SPH discretization

The full SPH discretization of the governing equations in our enhanced δ^+ -SPH framework with micropolar formulation combines all the previously discussed elements into a unified system: the fundamental SPH differential operators established in Section 2.1, the micropolar fluid dynamics from Section 2.2, and the complete set of stabilization techniques including δ -SPH density diffusion, tensile instability control (TIC), and our novel particle shifting technique (PST) from Section 2.3.

Continuity equation with diffusion

$$\frac{d\rho_i}{dt} = -\rho_i \sum_{j \in \mathcal{N}_i} (\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}) F_{ij} V_j + 2\delta h c_0 \sum_{j \in \mathcal{N}_i} \Psi_{ij} F_{ij} V_j, \quad (47)$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $V_j = m_j / \rho_j$, and Ψ_{ij} is the density diffusion term defined in Eq. (38).

Momentum equation with micropolar coupling

$$\begin{aligned} \frac{d\mathbf{v}_i}{dt} = & \mathbf{f}_i^p - \mathcal{K} \mathbf{v} \sum_{j \in \mathcal{N}_i} \pi_{ij} \mathbf{r}_{ij} F_{ij} V_j \\ & + (\nu + \nu_i) \sum_{j \in \mathcal{N}_i} (\mathbf{w}_i - \mathbf{w}_j) \times \mathbf{r}_{ij} F_{ij} V_j + \mathbf{g}, \end{aligned} \quad (48)$$

where $\mathcal{K} = 2(d+2)$, π_{ij} is defined in Eq. (29), and the pressure force $\mathbf{f}_i^p$ incorporates tensile instability control as described in Eq. (40).

Micro-rotation transport equation

$$\begin{aligned} \Theta \frac{d\mathbf{w}_i}{dt} = & -\mathcal{K} \xi \sum_{j \in \mathcal{N}_i} \Pi_{ij} \mathbf{r}_{ij} F_{ij} V_j \\ & + (\nu + \nu_i) \left(\sum_{j \in \mathcal{N}_i} (\mathbf{v}_i - \mathbf{v}_j) \times \mathbf{r}_{ij} F_{ij} V_j - 2\mathbf{w}_i \right) + \mathbf{f}_i, \end{aligned} \quad (49)$$

where Π_{ij} is defined in Eq. (35) and ξ is the angular viscosity coefficient.

Time integration with particle shifting

$$\begin{aligned} \mathbf{r}_i^{n+1} &= \mathbf{r}_i^n + \mathbf{v}_i^n \Delta t + \frac{1}{2} \frac{d\mathbf{v}_i^n}{dt} \Delta t^2 + \delta \mathbf{r}_i^{\text{final}}, \\ \mathbf{v}_i^{n+1} &= \mathbf{v}_i^n + \frac{d\mathbf{v}_i^n}{dt} \Delta t, \\ \mathbf{w}_i^{n+1} &= \mathbf{w}_i^n + \frac{d\mathbf{w}_i^n}{dt} \Delta t, \end{aligned} \quad (50)$$

where $\frac{d\mathbf{v}_i}{dt}$ and $\frac{d\mathbf{w}_i}{dt}$ are given by Eq. (48) and (49), respectively, and $\delta \mathbf{r}_i^{\text{final}}$ is the particle shift from Eq. (45).

This complete system integrates the micropolar fluid dynamics, δ^+ -SPH stabilization techniques, and advanced correction methods into a unified framework optimized for GPU implementation.

3. Novel GPU implementation strategy

3.1. Overview and motivation

The computational demands of SPH simulations have traditionally limited their application to offline calculations or small-scale problems. While GPU acceleration has been explored extensively, including shallow water SPH models [15], multi-GPU implementations on heterogeneous clusters [14], large-scale explicit incompressible SPH [16], CUDA-optimized weakly-compressible SPH [17], mixed-precision frameworks [19], and improved memory management strategies [18], these implementations predominantly rely on vendor-specific technologies such as CUDA, limiting portability across different GPU architectures.

To address this limitation, we present a GPU implementation leveraging OpenGL compute shaders that achieves exceptional performance

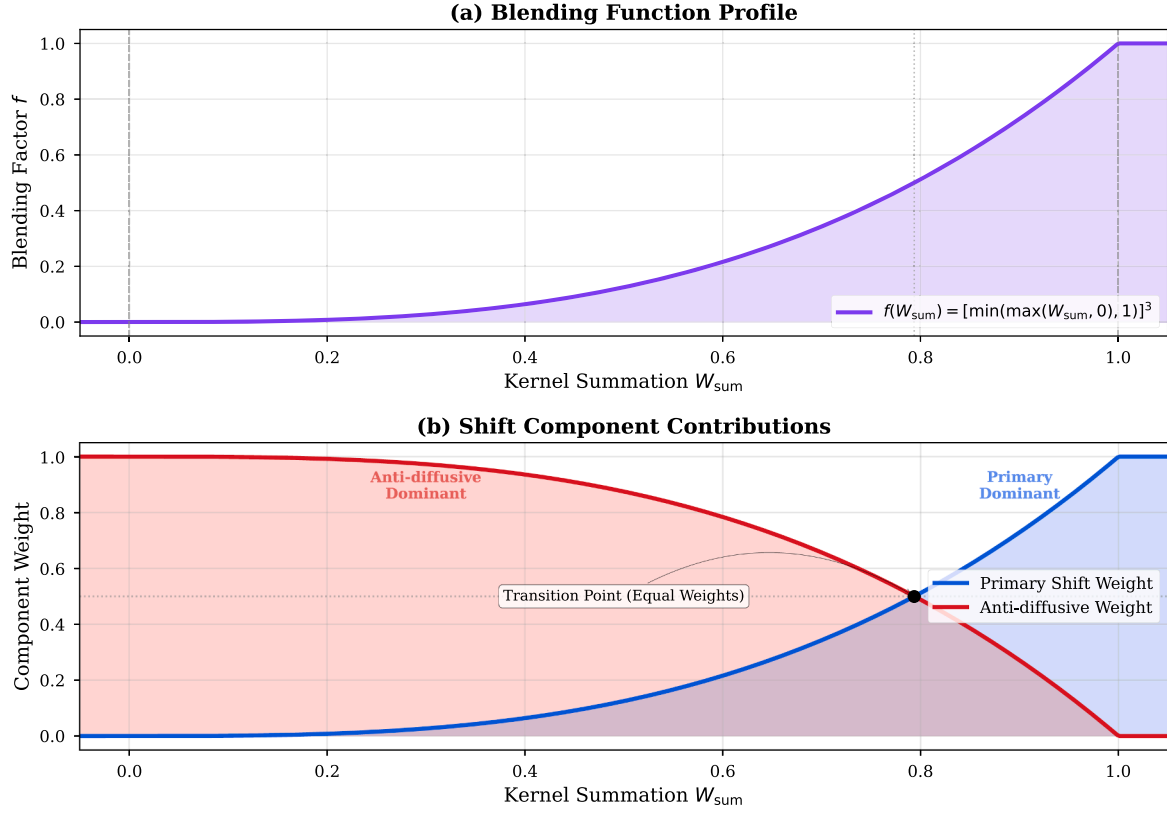


Fig. 3. Enhanced particle shifting technique with adaptive blending mechanism. (a) The cubic blending function profile showing smooth transition between regions. (b) Component weight distribution illustrating the complementary nature of primary and anti-diffusive shifts.

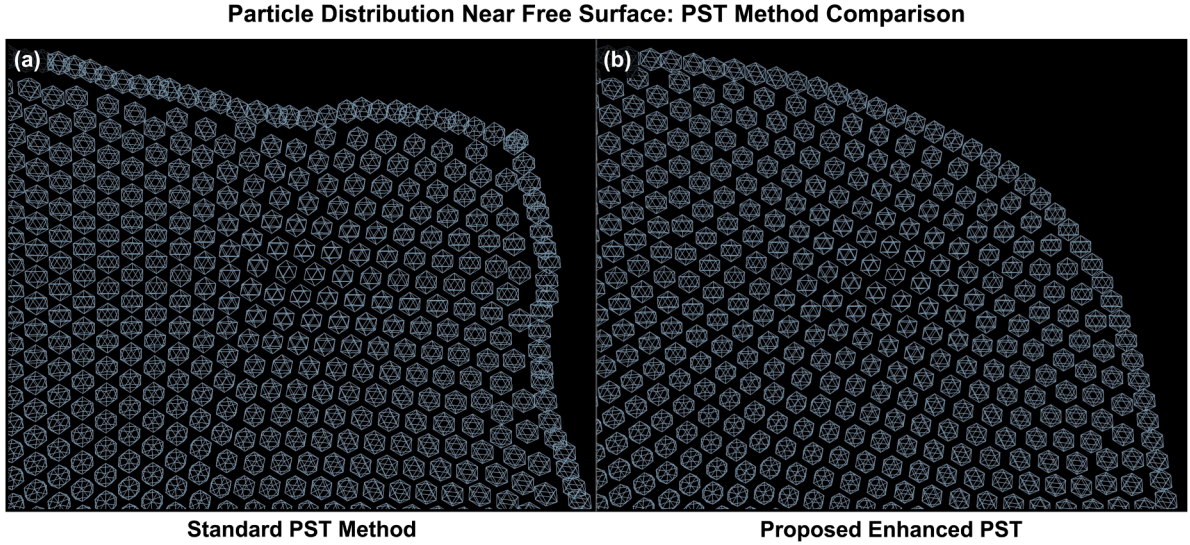


Fig. 4. Comparison of particle distributions near free-surface regions: conventional PST with Heaviside switching (a) versus our novel adaptive PST with continuous blending (b). Our method eliminates the issues of particle clustering and expensive free-surface detection, achieving superior particle regularity through innovative complementary weighting functions.

while maintaining cross-platform compatibility. Our approach is guided by three design principles: (1) hardware vendor independence through the standard OpenGL API, enabling deployment across NVIDIA, AMD, and Intel architectures without code modification; (2) seamless integration of computation and visualization via zero-copy memory architecture, eliminating I/O bottlenecks inherent in post-processing workflows; and (3) efficient memory management through hierarchical voxel struc-

tures with dynamic particle tracking, enabling million-particle simulations on consumer-grade hardware.

3.1.1. Distinction from prior GPU-SPH implementations

While existing GPU-accelerated SPH solvers such as Dual-SPHysics [14,20], GPUSPH [41], and the micropolar SPH implementation of Bender et al. and their solver SPlisHSPH [9,21] have

demonstrated significant performance improvements over CPU implementations, they share common architectural constraints that our framework addresses.

First, vendor lock-in remains a significant limitation. CUDA-based implementations, while highly optimized for NVIDIA hardware, cannot execute on AMD or Intel GPUs without substantial rewriting. Our OpenGL compute shader approach provides equivalent performance on NVIDIA hardware while enabling execution across all major GPU vendors, as demonstrated by our cross-platform validation on both NVIDIA RTX 5090 and AMD Radeon RX 7900 XTX (Section 3.4).

Second, most existing frameworks maintain architectural separation between simulation and visualization pipelines, requiring explicit data transfers for rendering. DualSPHysics, for example, exports particle data to disk for post-processing visualization. Our unified memory architecture maintains particle data exclusively in GPU memory throughout the simulation-visualization cycle, eliminating I/O bottlenecks and enabling true real-time interactive exploration.

Third, the degree of GPU utilization varies substantially among existing frameworks. SPLisHSPlasH [21], developed by Bender et al., implements state-of-the-art SPH formulations including micropolar fluids and provides both CompactNSearch for CPU-based neighborhood search and cuNSearch for GPU-accelerated neighborhood search via CUDA. However, the core SPH computations, including pressure solvers, viscosity evaluation, and time integration, remain CPU-based, with only the computationally expensive neighborhood search offloaded to the GPU. In contrast, our framework executes the entire simulation pipeline on the GPU, including all SPH operators, time integration, boundary handling, and particle shifting, achieving full GPU residency that eliminates CPU-GPU data transfers during the simulation loop. This architectural distinction enables the two-order-of-magnitude speedup reported in our benchmarks.

3.2. OpenGL compute shader architecture

3.2.1. Cross-platform design philosophy

Our framework employs OpenGL compute shaders rather than vendor-specific solutions such as CUDA or ROCm, ensuring compatibility across NVIDIA, AMD, and Intel GPU architectures. This cross-platform approach exploits OpenGL's mature driver infrastructure while providing seamless integration between compute and graphics pipelines, which is an essential requirement for real-time visualization. The OpenGL compute pipeline provides a key architectural advantage by maintaining unified memory space between computation and rendering stages, eliminating the CPU-GPU data transfers that typically bottleneck interactive applications. This zero-copy architecture enables real-time SPH simulation, a capability previously limited to offline processing. By adhering to open standards rather than vendor-owned proprietary technologies, our implementation ensures reproducibility and accessibility, allowing researchers to validate our results regardless of their hardware platform.

Here we present a detailed performance characterization using an NVIDIA RTX 5090 GPU. The framework achieves 300+ FPS with one million particles, demonstrating that cross-platform compatibility need not compromise performance. Fig. 5 demonstrates the solver's scaling behavior, achieving near-linear scaling across particle counts ranging from 0.25 to 32 million. The solver maintains computational throughput between 268 and 403 million particles per second, with optimal efficiency occurring at particle counts above 2 million, where throughput stabilizes above 370M particles/sec and peaks at 403M particles/sec. This performance profile validates our framework's ability to efficiently utilize modern GPU architectures while maintaining the flexibility of cross-platform deployment through standard OpenGL compute shaders.

3.2.2. Computational pipeline architecture

Our GPU computation pipeline consists of multiple specialized compute shaders organized in a cyclic execution pattern, as illustrated in Fig. 6. The pipeline employs a critical optimization strategy: we split the total computation into several sequential stages, where each stage precalculates and stores data required by subsequent stages as runtime variables. This approach minimizes expensive GPU kernel launches by consolidating computations and eliminating redundant calculations across stages.

The complete GPU workflow proceeds through the following carefully orchestrated phases:

Initialization Phase: The neighborhood register shader executes once during initialization to establish particle-voxel associations and populate the initial neighbor lists. This one-time setup provides the spatial indexing foundation for all subsequent computations.

Kernel Preprocessing Phase: In the first stage of each time step, we calculate and store three critical quantities for later use: the kernel-related values, the KGC matrix $\mathbf{L}$ described in Eq. (21) and the density gradients described in Eq. (39). These precomputed values are essential to ensure accurate SPH interpolations throughout the remaining computational stages.

Density and Pressure Update Phase: The second stage computes particle densities using the δ -SPH continuity Eq. (47), leveraging the precalculated kernel function data from the previous phase. The pressure field is then immediately evaluated using the Tait equation of state (31).

Force Computation Phase: The third stage calculates particle dynamics based on the selected physical model. For standard fluid simulations, we solve the Navier-Stokes Eq. (30), while micropolar fluids utilize the extended micropolar formulation (48) and (49). This shader evaluates pressure gradients, viscous forces, and any additional forces such as curl-based vorticity confinement or transfer forces. When PST is enabled, the shifting displacements defined in Eq. (45) are also calculated during this phase.

Time Integration Phase: The fourth stage performs temporal integration to update particle positions, velocities, and, in the case of micropolar fluids, micro-rotation vectors. During this phase, we also maintain voxel in/out buffers that track particle boundary crossings. Whenever a particle's updated position places it outside its current voxel, we record both the departure from the original voxel and the entry into the destination voxel.

Spatial Indexing Update Phase: The final voxel shader updates the particle-voxel associations and updates the neighborhood lists based on the in/out buffers. This updated spatial relationship then feeds into the next time step's kernel preprocessing phase, completing the cyclic execution pattern.

This staged architecture achieves full GPU residency by storing all intermediate quantities in shader storage buffer objects (SSBOs) that persist across shader dispatches, thereby eliminating CPU-GPU memory transfers during the simulation loop. The precalculation strategy reduces the total number of kernel dispatches per timestep from what would otherwise require separate passes for each SPH operator, while ensuring that expensive quantities such as the KGC matrix $\mathbf{L}$ are computed only once per particle per timestep. The performance implications of this design are quantified in Section 3.4, where direct comparison against DualSPHysics demonstrates a 17–18% reduction in wall-clock time on equivalent GPU hardware.

3.2.3. Real-time visualization integration

A distinguishing feature of our implementation is the seamless integration between computation and visualization through a unified GPU memory architecture. Traditional SPH implementations suffer from expensive CPU-GPU data transfers that often become the primary bottleneck in interactive applications. Our framework eliminates this overhead by maintaining all particle data within GPU memory throughout the entire simulation-visualization pipeline, as illustrated in Fig. 7.

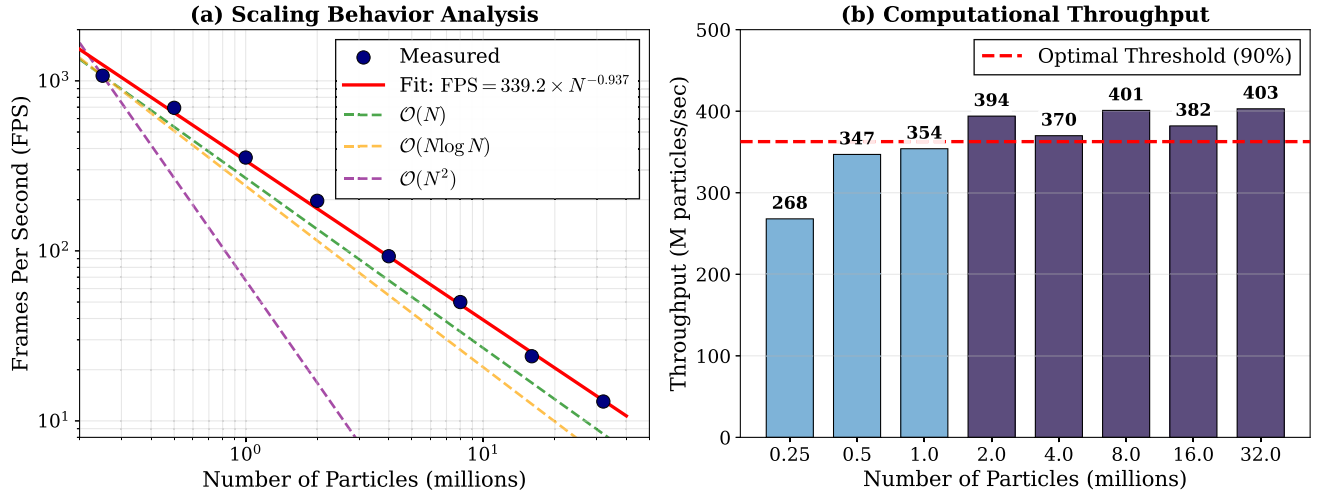


Fig. 5. SPH solver performance scaling on NVIDIA RTX 5090. (a) Scaling behavior analysis showing measured frame rates with min-max ranges and fitted power law compared to theoretical complexity bounds. (b) Computational throughput in million particles per second across different particle counts, with the optimal threshold (90% of peak) indicated at 354 M particles/sec. Peak throughput of 403 M particles/sec is achieved at 32 million particles.

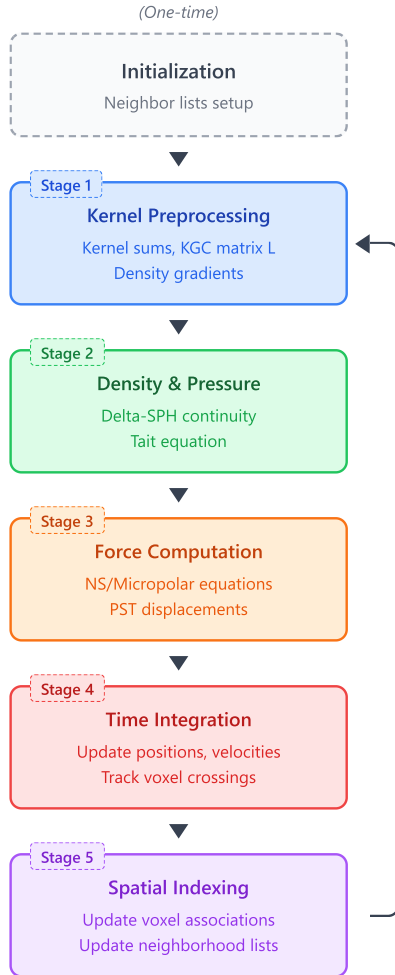


Fig. 6. GPU computation pipeline with five cyclic stages executing specialized compute shaders. After one-time initialization, the pipeline sequentially processes kernel preprocessing, density/pressure updates, force computation, time integration, and spatial indexing, with precalculated data stored in GPU memory to minimize kernel launch overhead.

This zero-copy architecture achieves true real-time visualization through three key design elements. First, the **shared SSBO memory pool** maintains particle data permanently in GPU memory as a shared shader storage buffer object, accessible to both compute and rendering shaders without intermediate transfers. After the initial CPU-to-GPU transfer during setup, all operations from neighborhood searches through force calculations to final rendering operate on this shared memory pool, eliminating the gigabytes-per-second transfer overhead that would otherwise be required for million-particle simulations. Second, **asynchronous pipeline execution** allows the rendering pipeline to operate independently from the compute loop with read-only access to the shared SSBO, enabling users to interactively adjust visualization parameters and camera positions while computations proceed uninterrupted. Third, the **flexible operation modes** support dynamic switching between real-time visualization for interactive exploration and headless computation for maximum performance during production runs.

When detailed analysis is required, simulation data can be exported to the CPU for post-processing tasks such as statistical analysis, data export to standard formats, or offline rendering.

3.3. Voxel-based spatial data structure

3.3.1. Four component voxel architecture

Efficient neighbor searching is fundamental to SPH performance, as each particle queries approximately 50–100 neighbors per timestep in three-dimensional simulations. Our voxel-based spatial subdivision addresses this computational bottleneck through a four-component architecture specifically optimized for GPU memory access patterns. These four components consist of a voxel header storing spatial metadata and precomputed neighborhood connectivity, an inside buffer maintaining the current particle list within the voxel, an in-buffer tracking particles entering the voxel, and an out-buffer recording particles exiting the voxel. This decomposition enables lock-free parallel updates while maintaining spatial coherency.

The voxel header stores two categories of spatial metadata. The first category comprises identity information: a unique voxel identifier and a 3D grid offset vector specifying the voxel's position within the computational domain. The second category contains a precomputed 26-connectivity map that indexes all face-adjacent (6), edge-adjacent (12), and corner-adjacent (8) neighboring voxels. This precomputation enables $\mathcal{O}(1)$ neighbor voxel lookup without runtime spatial calculations, critical for GPU efficiency. Fig. 8 illustrates this header structure.

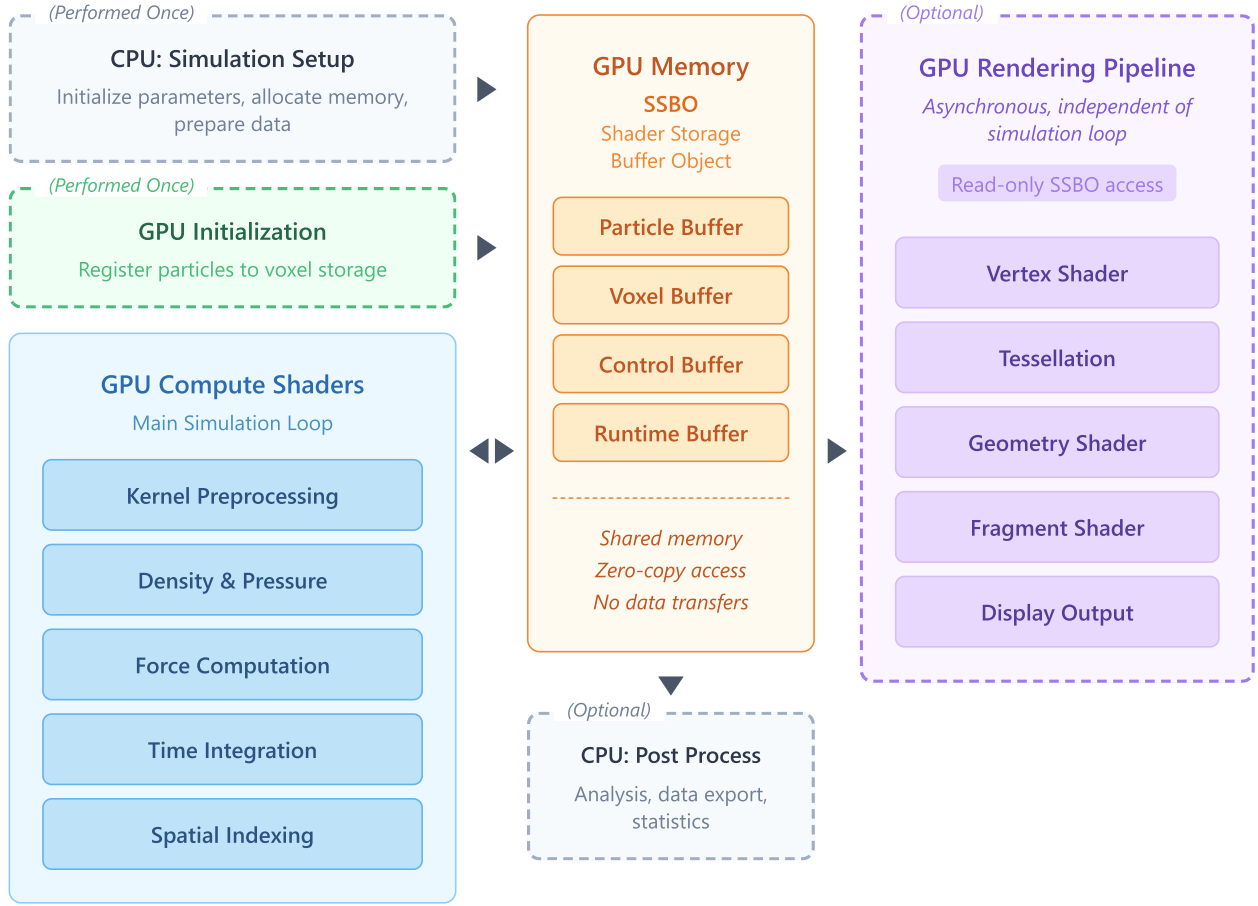


Fig. 7. GPU-accelerated SPH pipeline architecture with unified memory design. The central SSBO provides zero-copy shared memory access between the compute shaders (left) executing the five-stage simulation loop and the rendering pipeline (right) performing real-time visualization. After initial CPU-to-GPU transfer, all operations occur in GPU memory with asynchronous execution of compute and rendering pipelines, eliminating traditional data transfer bottlenecks.

Voxel Identity Information ►

Identifier	x_offset	y_offset	z_offset
Left	Right	Down	Up
Back	Front	LeftDown	LeftUp
RightDown	RightUp	LeftBack	LeftFront
RightBack	RightFront	DownBack	DownFront
UpBack	UpFront	LeftDownBack	LeftDownFront
LeftUpBack	LeftUpFront	RightDownBack	RightDownFront
RightUpBack	RightUpFront	0	0

Voxel Spatial Information ►

Fig. 8. Hierarchical voxel data structure showing identity information with unique identifier and 3D offset coordinates, and precomputed 26-connectivity neighbor indices for $\mathcal{O}(1)$ spatial queries.

In addition to the voxel header, each voxel maintains three particle index buffers, completing the four-component architecture: an inside buffer storing current particle indices, and paired in/out buffers tracking particle migrations. The inside buffer enables lock-free neighbor queries by maintaining a fixed-size list of particles within the voxel boundaries. During position updates, particles crossing the voxel boundaries are recorded in the outgoing buffer of the source voxel and the incoming buffer of the destination voxel using atomic operations. This paired in/out buffer approach decouples read and write operations during particle redistribution, preventing race conditions while maintaining parallel efficiency. The separation of migration tracking from the main particle list eliminates complex synchronization that would otherwise limit GPU throughput.

3.3.2. Dynamic particle tracking mechanism

The in-out buffer system addresses critical memory access conflicts that arise during parallel particle migration by enabling efficient inter-voxel particle movement tracking without expensive global synchronization operations. This mechanism, detailed in Algorithm 2, decouples the read and write operations during particle redistribution, preventing race conditions when multiple threads attempt to modify the same voxel's particle list simultaneously.

Algorithm 2 In-out buffer update algorithm.

Stage 1: Movement Detection

```

for each particle  $i$  in parallel do
    Compute new voxel index from updated position
    if voxel changed then
        Add particle to destination voxel's in-buffer
        Add particle to source voxel's out-buffer
    end if
end for

```

Stage 2: Buffer Reconciliation

```

for each voxel  $v$  in parallel do
    Remove out-buffer particles from inside buffer
    Add in-buffer particles to inside buffer
    Validate particle count consistency
    Clear in/out buffers for next iteration
end for

```

This architecture provides several computational advantages that directly address GPU memory access patterns. By allowing each voxel to independently manage its own particle lists, the system eliminates the need for global synchronization barriers that would otherwise stall parallel execution. The approach maintains $\mathcal{O}(N)$ complexity for neighbor searching operations, ensuring linear scaling with particle count even in dense configurations. Most importantly, the separation of in-buffer and out-buffer operations reduces memory contention by staging particle migrations in temporary buffers rather than directly modifying the main particle lists. While atomic operations are still required for appending to these temporary buffers, the two-phase approach prevents the more complex synchronization issues that arise when multiple threads attempt to simultaneously insert and remove particles from the main voxel lists. During the reconciliation phase, each voxel independently processes its own buffers without inter-voxel synchronization, eliminating global barriers. This design choice proves particularly effective in scenarios with high particle mobility, where traditional direct-update approaches would require complex locking mechanisms or suffer from race conditions when modifying shared particle lists.

3.3.3. Cache-optimized particle resorting

Modern GPU architectures exhibit significant performance variations based on memory access patterns, with coalesced memory accesses achieving much higher throughput than scattered accesses [42]. Although our voxel-based spatial structure ensures efficient neighborhood detection, the dynamic nature of particle movements inevitably

leads to memory fragmentation over time. Particles that were initially contiguous in memory become scattered as they migrate between voxels during simulation, which occurs much more rapidly with vortical flow, degrading cache efficiency and memory bandwidth utilization. To address this performance degradation, we implement a periodic particle sorting Algorithm 3 that reorganizes the particle indices to restore spatial locality in memory, as shown in Fig. 9.

The fundamental observation underlying our approach is that particles within the same voxel interact frequently during SPH computations, both in neighbor searches and in force calculations. By ensuring these frequently-interacting particles have contiguous memory addresses, we enhance L1/L2 cache utilization and promote coalesced memory accesses within GPU warps. This spatial-temporal locality optimization proves particularly effective for SPH's computational pattern, where each particle repeatedly accesses the properties of its neighbors within a localized spatial region.

The performance impact of our cache-optimized resorting strategy is demonstrated in Fig. 10. On the AMD Radeon RX 7900 XTX GPU, which features 96 MB of Infinity Cache and relies heavily on cache hierarchy for performance, maintaining spatial locality proves critical. Without periodic resorting, the simulation experiences continuous performance degradation, declining from an initial 130 FPS to approximately 85 FPS over 80 seconds of execution time, leading to a 35% performance loss. This degradation stems from increasingly scattered memory access patterns as particles migrate through the fluid domain, causing cache misses and inefficient memory bandwidth utilization. In contrast, with our resorting algorithm applied periodically, the system maintains stable performance at 114.0 ± 6.0 FPS, representing a 17% improvement over the non-resorted case's average of 97.4 ± 11.0 FPS. The periodic spikes visible in the resorted performance correspond to the resorting operations themselves, which incur minimal overhead while preventing the accumulation of memory fragmentation. This stability is particularly crucial for real-time applications where consistent frame rates are essential for user experience.

Algorithm 3 Cache-optimized particle resorting algorithm.

Require: Current particle buffers, voxel buffers

Ensure: Spatially contiguous particle indexing within voxels

```

Initialize new empty particle buffers with same shape as original particle buffers
for all voxels do
    for all particles in voxel do
        Copy particle to new particle buffers with new index
        Update particle index in voxel
    end for
    Write updated voxel back to voxel buffer
end for

```

3.3.4. Comparison with existing cell-based methods

Background cell methods with particle sorting for memory locality enhancement are well-established in GPU-accelerated SPH [16,42]. In these approaches, particle indices are sorted according to cell membership during each timestep, with particle data redistributed to ensure spatially proximate particles occupy contiguous memory addresses. For instance, Morikawa et al. [16] employ Thrust's `sort_by_key` to reorder all particles by cell index during each neighbor search operation. Rather than maintaining explicit particle lists per cell, this approach stores only the first and last particle indices for each cell, relying on the sorted contiguity of the particle array to implicitly define cell membership. Consequently, the cell-particle associations must be reconstructed through global sorting whenever particles move.

Our implementation differs from these sort-based approaches in two key aspects. First, our voxel-based spatial structure maintains explicit particle index lists within each voxel, initialized once at the beginning

Cache-Optimized Particle Resorting



Fig. 9. One example of Cache-optimized particle resorting mechanism. The voxel structure maintains three components: spatial metadata saved in header, boundary particles with negative indices, and fluid particles. During resorting, only fluid particles are reassigned from scattered indices to contiguous sequences, while header and boundary data remain unchanged. This reorganization ensures particles within the same voxel occupy consecutive memory addresses, improving cache locality.

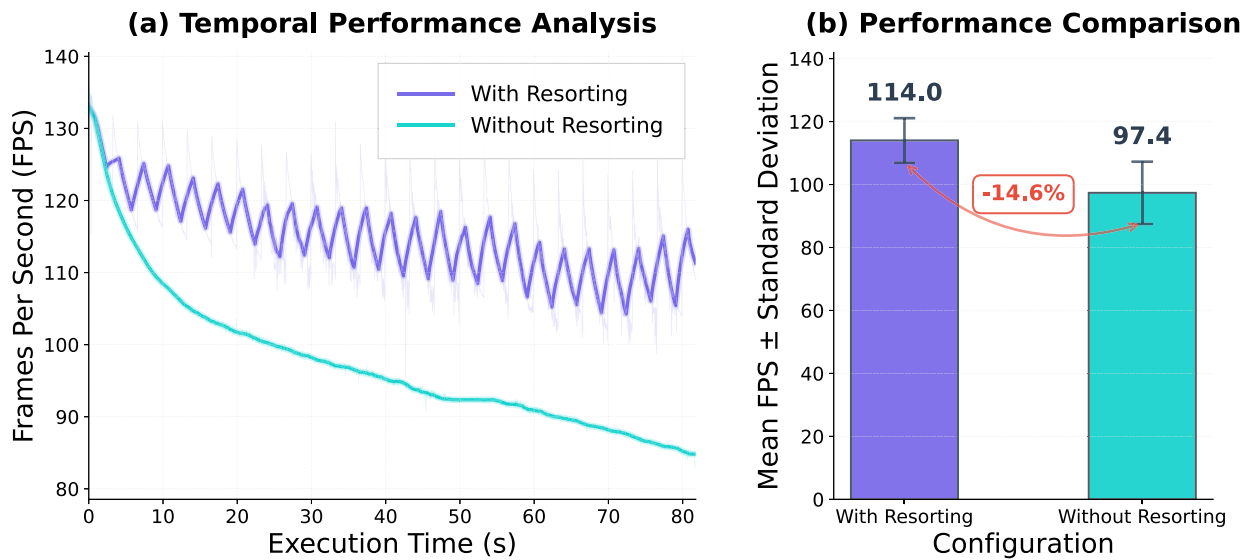


Fig. 10. Performance impact of cache-optimized particle resorting on AMD Radeon RX 7900 XTX GPU. (a) Temporal performance comparison showing FPS over execution time with a 100-frame moving average. Without resorting, performance degrades continuously from 130 to 85 FPS due to memory fragmentation. With periodic resorting every 100 timesteps, stable performance is maintained around 114 FPS. (b) Overall performance comparison showing 17% improvement in mean FPS and significantly reduced variance with resorting enabled.

of the simulation and updated incrementally through the in-buffer/out-buffer mechanism. This explicit per-voxel storage eliminates the need for global sorting to maintain cell-particle associations, as particle migration is tracked locally through atomic append operations to per-voxel temporary buffers. During the reconciliation phase, each voxel independently processes its own buffers to update its particle list, enabling parallel updates without the collective particle reordering inherent in sort-based methods. Second, similar to [16], we do not maintain precomputed neighbor lists; instead, neighbors are identified by searching particles within the current voxel and its 26 adjacent voxels during each force computation. While this requires distance calculations for all candidate particles, modern GPU architectures favor such computation-bound local arithmetic operations over the memory-bound global transactions and synchronization overhead associated with per-timestep sorting. Memory locality is maintained through periodic resorting (Section 3.3.3) performed only when accumulated fragmentation causes detectable performance degradation, rather than through per-timestep global sorting. The performance implications of this design choice are quantified in Section 3.4.

3.4. Performance benchmarks and comparison

To quantify the computational efficiency of our implementation, we conducted systematic benchmarks comparing against DualSPHysics v5.4 [12], the most widely-used open-source GPU-accelerated SPH solver. Tests were performed on three systems with different CPU and GPU configurations to evaluate both performance and cross-platform compatibility.

3.4.1. Benchmark results

Two standard dam break test cases from the DualSPHysics examples were selected: a 2D case with 21,001 particles and a 3D case with 171,496 particles. Both solvers were configured with identical physical parameters and time step sizes to ensure a fair comparison.

The benchmarks were executed on three systems: **System A** equipped with an AMD Ryzen 9 9950X CPU (16 cores) and NVIDIA RTX 5090 GPU, **System B** with an AMD Ryzen 7 9800X3D CPU (8 cores) and AMD RX 7900 XTX GPU, and **System C** with an Intel Xeon Gold 6226R CPU (16 cores) and NVIDIA RTX 5090 GPU. Systems A and B represent high-end consumer configurations, while System C represents a typical server-class workstation. Table 1 summarizes the results.

On System A, the present work achieves wall times of 39.1 s and 20.6 s for the 2D and 3D cases respectively, compared to 47.6 s and 24.8 s for DualSPHysics GPU solver. This corresponds to approximately 17–18% reduction in computation time relative to DualSPHysics GPU. The performance improvement can be attributed to two key design choices. First, the in-out buffer system enables efficient inter-voxel particle movement tracking without expensive global synchronization operations, allowing all modifications to the neighborhood voxel buffer to be fully parallelized. Second, periodic particle re-indexing is performed to maintain memory access locality. This resorting operation consumes only approximately 5% of the total computation time, yet effectively preserves cache efficiency throughout the simulation, preventing performance degradation caused by non-contiguous memory access patterns as particles become spatially disordered over time.

The cross-platform capability of the OpenGL-based implementation is demonstrated on System B, where the same code executes on an AMD GPU without modification. DualSPHysics GPU solver cannot run on AMD hardware due to its CUDA dependency. On the RX 7900 XTX, the present work achieves speedups of 3.0× and 40.7× over the CPU baseline for the 2D and 3D cases respectively, demonstrating that the OpenGL compute shader implementation effectively utilizes GPU resources across different hardware vendors.

On System C, which represents a server-class workstation configuration, the speedup relative to CPU is even more pronounced. The Intel Xeon Gold 6226R requires 2339.6 s for the 3D case, whereas the present

work completes in 24.1 s, achieving a 97× speedup that approaches two orders of magnitude. This substantial improvement highlights the particular benefit of GPU acceleration for HPC environments where server-class CPUs, despite their high core counts, cannot match the massive parallelism of modern GPUs for particle-based simulations.

3.5. Discussion and limitations

While our GPU implementation demonstrates significant performance improvements, several limitations warrant consideration.

The current framework employs a single GPU architecture with all particle data residing in unified device memory. This design maximizes performance through zero-copy memory access and eliminates synchronization overhead within a single device. However, extending to multiple GPU configurations would require substantial architectural modifications to handle inter-device communication and workload distribution. The PCIe bandwidth limitations and synchronization costs between GPUs present non-trivial challenges for maintaining the observed performance scaling across multiple devices.

Additionally, applications requiring double-precision arithmetic encounter hardware-imposed constraints on consumer GPUs, where FP64 throughput is typically 1/32 to 1/64 of FP32 performance. While our framework supports double precision, users must accept reduced frame rates for simulations demanding higher numerical accuracy. This limitation is inherent to consumer GPU architectures optimized for single-precision graphics workloads rather than scientific computing.

Memory capacity also constrains maximum problem size. Although our implementation efficiently handles 32 million particles within 12 GB of GPU memory, industrial-scale simulations requiring hundreds of millions of particles exceed the memory capacity of current consumer GPUs. Professional GPUs with larger memory capacities could address this limitation but at significantly higher cost.

Nevertheless, the achieved performance levels enable real-time interactive exploration of fluid phenomena that previously required offline batch processing, transforming hours of computation into immediate visual feedback, facilitating rapid parameter studies and design iteration within practical hardware constraints.

4. Numerical experiments and results

Having established the computational framework and GPU implementation strategy, we now validate the enhanced δ^+ -SPH formulation through three test cases of progressively increasing complexity. The lid-driven cavity examines confined viscous flows and vortex dynamics at Reynolds numbers from 250 to 100,000, including micropolar fluid validation. Flow around an elliptical cylinder (aspect ratio 2:1) tests external flow phenomena such as boundary layer development, vortex shedding, and the drag crisis at Reynolds numbers up to 10^6 . Finally, NACA airfoil simulations (NACA0008, NACA0015, and NACA6409) assess aerodynamic performance including stall behavior at Reynolds numbers up to 2×10^5 . Each case exploits GPU acceleration to perform high-resolution simulations with millions of particles, validated against established benchmarks, numerical reference solutions, and experimental data.

4.1. Lid driven cavity validation

4.1.1. Problem description

The lid driven cavity flow represents one of the most fundamental benchmark problems in computational fluid dynamics, serving as a standard validation case since the pioneering work of Ghia et al. [43]. This canonical problem consists of a square cavity with three stationary walls and a top wall moving at constant velocity. Despite its geometric simplicity, the lid-driven cavity exhibits rich flow physics, including primary and secondary vortex formation, corner singularities, and increasingly complex vortical structures with rising Reynolds number. These

Table 1

Performance comparison between DualSPHysics v5.4 and the present work on standard dam break benchmarks. Both solvers use identical physical setups with $CFL = 0.2$. Speedup is calculated relative to DualSPHysics CPU on each system. The present work uses OpenGL compute shaders, enabling execution on both NVIDIA and AMD GPUs, whereas DualSPHysics GPU only supports NVIDIA hardware.

Case	Solver	Hardware	Particles	Wall Time (s)	Speedup
<i>System A: AMD Ryzen 9 9950X (16 cores) + NVIDIA RTX 5090</i>					
2D Dam Break ($T = 2.0$ s)	DualSPHysics CPU	9950X	21,001	116.2	1.0×
	DualSPHysics GPU	RTX 5090	21,001	47.6	2.4×
	Present work	RTX 5090	21,001	39.1	3.0×
3D Dam Break ($T = 1.6$ s)	DualSPHysics CPU	9950X	171,496	760.3	1.0×
	DualSPHysics GPU	RTX 5090	171,496	24.8	30.6×
	Present work	RTX 5090	171,496	20.6	36.9×
<i>System B: AMD Ryzen 7 9800X3D (8 cores) + AMD RX 7900 XTX</i>					
2D Dam Break ($T = 2.0$ s)	DualSPHysics CPU	9800X3D	21,001	125.0	1.0×
	DualSPHysics GPU	RX 7900 XTX	N/A (CUDA required)	—	—
	Present work	RX 7900 XTX	21,001	42.3	3.0×
3D Dam Break ($T = 1.6$ s)	DualSPHysics CPU	9800X3D	171,496	1324.1	1.0×
	DualSPHysics GPU	RX 7900 XTX	N/A (CUDA required)	—	—
	Present work	RX 7900 XTX	171,496	32.5	40.7×
<i>System C: Intel Xeon Gold 6226R (16 cores) + NVIDIA RTX 5090</i>					
2D Dam Break ($T = 2.0$ s)	DualSPHysics CPU	Xeon 6226R	21,001	240.8	1.0×
	DualSPHysics GPU	RTX 5090	21,001	58.5	4.1×
	Present work	RTX 5090	21,001	48.4	5.0×
3D Dam Break ($T = 1.6$ s)	DualSPHysics CPU	Xeon 6226R	171,496	2339.6	1.0×
	DualSPHysics GPU	RTX 5090	171,496	28.6	81.8×
	Present work	RTX 5090	171,496	24.1	97.1×

characteristics make it an ideal test case for evaluating the capability of our enhanced δ^+ -SPH formulation to accurately capture rotational flows and fine-scale structures.

We simulate flow within a two-dimensional square cavity with edge length $L = 0.1$ m. The upper wall translates horizontally at constant velocity $U = 1.0$ m/s, while the remaining three walls remain stationary with no-slip boundary conditions. The flow behavior is characterized by the Reynolds number:

$$Re = \frac{UL}{\nu}, \quad (51)$$

where ν denotes the kinematic viscosity. Our investigation proceeds in two stages: first, quantitative validation at $Re \in \{100, 400, 1000, 3200, 7500, 10000\}$ against the benchmark data of Ghia et al. [43] and Marchi et al. [44]; second, extended simulations at $Re \in \{250, 500, 1250, 2500, 5000, 12500, 25000, 100000\}$ to examine flow physics across the laminar-to-chaotic transition. Additionally, micropolar fluid simulations at $Re = 400$ with varying vortex viscosity parameters validate the implementation of the extended constitutive model.

4.1.2. Numerical configuration

The simulations employ the enhanced δ^+ -SPH formulation with GPU acceleration as described in Section 3. Table 2 summarizes the key parameters for the baseline configuration with one million particles.

The reference sound speed $c_0 = 150$ m/s yields a Mach number $Ma = U/c_0 < 0.01$, ensuring density variations remain within 1% as required by the weakly compressible assumption. The Reynolds number is varied by adjusting the kinematic viscosity ν according to Eq. (51).

All simulations initialize from a quiescent state with zero velocity and micro-rotation fields. Particle densities are set to ρ_0 with uniform pressure throughout the domain. The adaptive time-stepping scheme based on the CFL criterion ensures numerical stability.

4.1.3. Quantitative validation against benchmark data

To validate the accuracy of our enhanced δ^+ -SPH formulation, we conduct simulations at $Re \in \{100, 400, 1000, 3200, 7500, 10000\}$ and com-

Table 2

Simulation parameters for the lid-driven cavity problem.

Parameter	Symbol	Value
Domain length	L	0.1 m
Lid velocity	U	1.0 m/s
Number of fluid particles	N	1,000,000
Resolution parameter	$n = L/\Delta x$	1000
Initial particle spacing	Δx	1×10^{-4} m
Smoothing length	h	5×10^{-4} m
CFL number	—	0.15
Reference sound speed	c_0	150 m/s
Tait equation exponent	γ	7
Reference density	ρ_0	1000 kg/m ³

pare velocity profiles along the cavity centerlines with the benchmark data of Ghia et al. [43] and the high-resolution finite difference solutions of Marchi et al. [44].

Fig. 11 presents the horizontal velocity u/U along the vertical centerline ($x = 0.5$) and the vertical velocity v/U along the horizontal centerline ($y = 0.5$). The SPH results demonstrate excellent agreement with both benchmark datasets across all Reynolds numbers. At low Reynolds numbers ($Re = 100$ and 400), the characteristic S-shaped velocity profiles are accurately captured. As Reynolds number increases, the velocity gradients near the walls steepen due to thinner boundary layers, and the method successfully resolves these increasingly sharp profiles without introducing excessive numerical diffusion. At the highest Reynolds numbers ($Re = 7500$ and 10000), where the velocity drops rapidly from unity at the lid to negative values within a thin shear layer, the agreement remains excellent.

Table 3 summarizes the relative L_2 errors, which remain below 1.3% across all cases. Detailed velocity data are provided in Appendix A.

4.1.4. Spatial convergence study

To assess spatial convergence, we perform simulations at $Re = 400$ using three resolutions: $n = L/\Delta x \in \{250, 500, 1000\}$, corresponding to

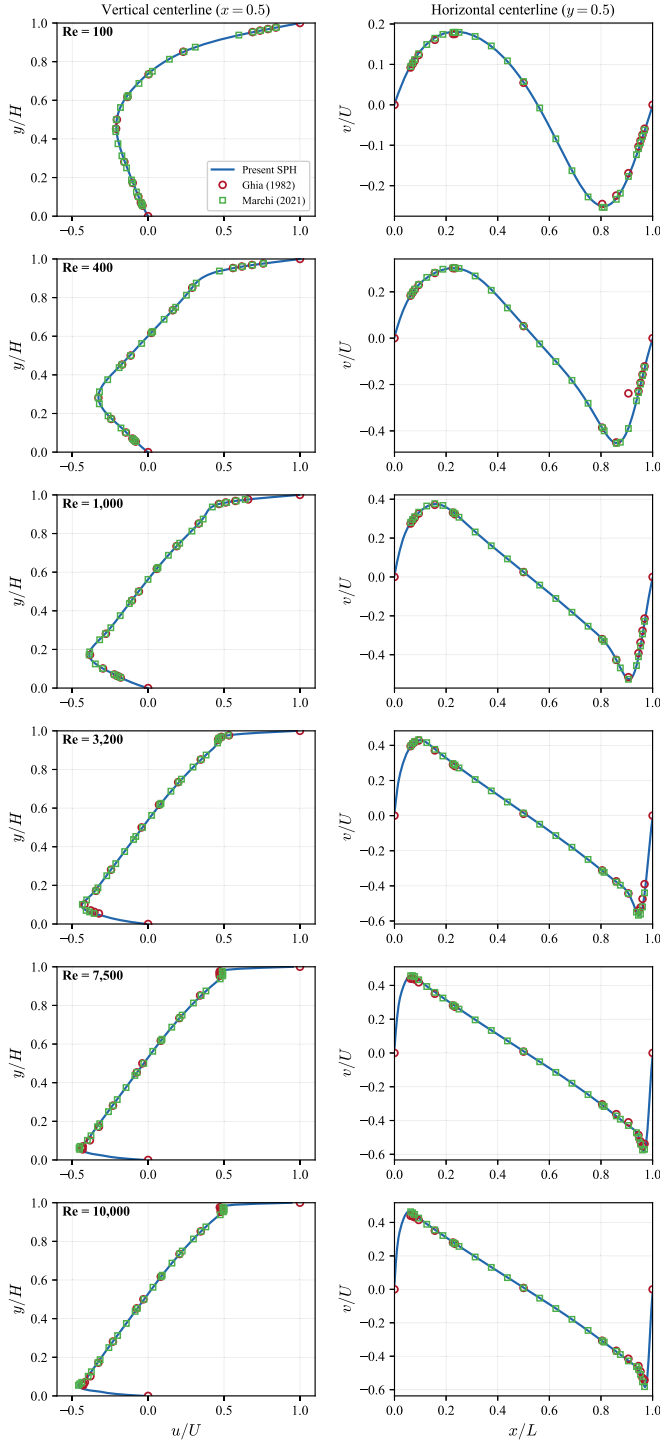


Fig. 11. Velocity profiles along cavity centerlines for Reynolds numbers $Re = 100, 400, 1000, 3200, 7500$, and 10000 : (left) horizontal velocity u/U along the vertical centerline ($x = 0.5$); (right) vertical velocity v/U along the horizontal centerline ($y = 0.5$). Present SPH results (solid lines) are compared with benchmark data from Ghia et al. [43] (circles) and Marchi et al. [44] (squares).

particle counts of $N \approx 6.25 \times 10^4$, 2.5×10^5 , and 1.0×10^6 , respectively. Fig. 12 compares the velocity profiles with the benchmark data of Marchi et al. [44], showing progressive improvement with increasing resolution.

Table 4 summarizes the error norms. The relative L_2 errors decrease from 5.80% to 0.93% for the u -velocity and from 2.98% to 0.80% for the v -velocity as resolution increases from $n = 250$ to $n = 1000$. Assum-

Table 3

Relative L_2 errors (%) compared with Marchi et al. [44].

Re	u (vertical centerline)	v (horizontal centerline)
100	1.21	1.05
400	0.93	0.80
1000	1.27	0.70
3200	0.93	0.71
7500	0.81	0.68
10,000	0.59	0.65

Table 4

Spatial convergence of velocity errors at $Re = 400$.

u -velocity (vertical centerline)				
n	ϵ_{L_1}	ϵ_{L_2}	ϵ_{L_∞}	$\epsilon_{L_2}^{\text{rel}} (\%)$
250	1.23×10^{-2}	1.89×10^{-2}	5.01×10^{-2}	5.80
500	7.88×10^{-3}	9.41×10^{-3}	2.12×10^{-2}	2.89
1000	2.62×10^{-3}	3.03×10^{-3}	5.84×10^{-3}	0.93
v -velocity (horizontal centerline)				
n	ϵ_{L_1}	ϵ_{L_2}	ϵ_{L_∞}	$\epsilon_{L_2}^{\text{rel}} (\%)$
250	5.90×10^{-3}	8.00×10^{-3}	1.95×10^{-2}	2.98
500	4.60×10^{-3}	5.12×10^{-3}	8.29×10^{-3}	1.91
1000	1.94×10^{-3}	2.15×10^{-3}	3.97×10^{-3}	0.80

ing a power-law relationship $\epsilon \propto (\Delta x)^p$, the estimated convergence rates between $n = 500$ and $n = 1000$ are:

$$p_u \approx \frac{\ln(\epsilon_{L_2,500}^u / \epsilon_{L_2,1000}^u)}{\ln 2} \approx 1.6, \quad p_v \approx \frac{\ln(\epsilon_{L_2,500}^v / \epsilon_{L_2,1000}^v)}{\ln 2} \approx 1.3. \quad (52)$$

These rates fall within the typical range of 1–2 reported for SPH schemes with fixed $h/\Delta x$ ratios [25], confirming reliable convergence behavior.

4.1.5. Flow physics at high reynolds numbers

Figs. 13 and 14 present the temporal evolution of vorticity and velocity magnitude fields for Reynolds numbers from 250 to 25,000. The simulations capture the systematic progression from steady laminar flow to complex multi-vortex structures. In the laminar regime ($Re \leq 500$), the flow rapidly develops into a steady configuration dominated by a single primary vortex, with small secondary vortices in the bottom corners reaching equilibrium within $t = 1.0$ s.

As Reynolds number increases to the transitional regime ($Re = 1250$ –5000), the primary vortex strengthens while secondary corner vortices intensify. At $Re = 2500$ and 5000, tertiary vortices emerge in the upper-left corner and the convergence time extends to $t \approx 2$ –3s as competing vortical structures establish equilibrium.

At high Reynolds numbers ($Re \geq 12500$), the flow exhibits increasingly complex dynamics. The primary vortex undergoes significant deformation and interacts with intensifying corner vortices to form a network of multi-scale structures. At $Re = 25000$, the primary vortex fragments into multiple interacting vortices, with chaotic behavior particularly pronounced in regions of strong shear. The velocity fields confirm these observations, showing smooth contours at low Reynolds numbers that progressively develop irregular, time-varying structures.

Fig. 15 presents the case at $Re = 100000$, where fully chaotic dynamics emerge. Extensive fine-scale vorticity structures develop within the first second. A cascade of two-dimensional vortices spanning multiple length scales persists through $t = 20.0$ s without converging to a steady or periodic state. This successful capture of complex flow characteristics at $Re = 100000$ demonstrates the capability of the enhanced δ^+ -SPH formulation to preserve essential physics throughout the laminar-to-chaotic transition.

4.1.6. Micropolar fluid validation

To validate the micropolar fluid implementation, we follow the approach demonstrated by Souto-Iglesias et al. [37] and adopt the formulation of Venkatadri et al. [45]:

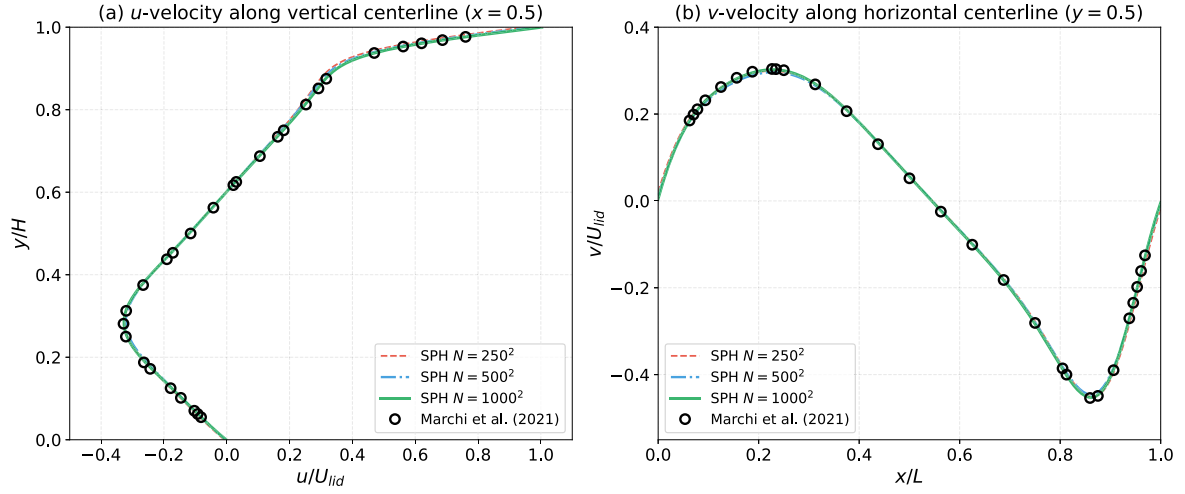


Fig. 12. Spatial convergence study at $Re = 400$: (a) horizontal velocity u/U along the vertical centerline; (b) vertical velocity v/U along the horizontal centerline. SPH results for $n \in \{250, 500, 1000\}$ are compared with Marchi et al. [44] (circles).

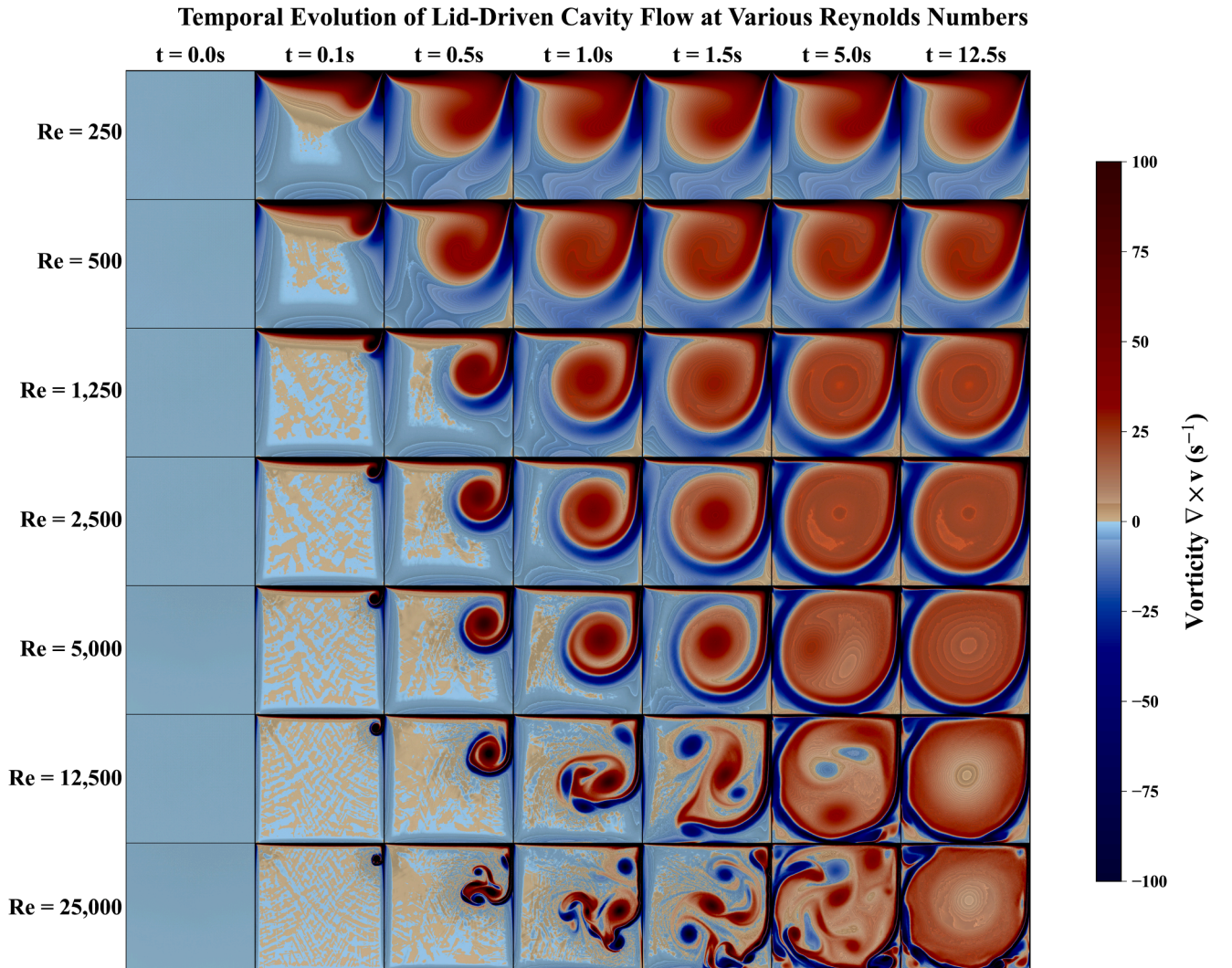
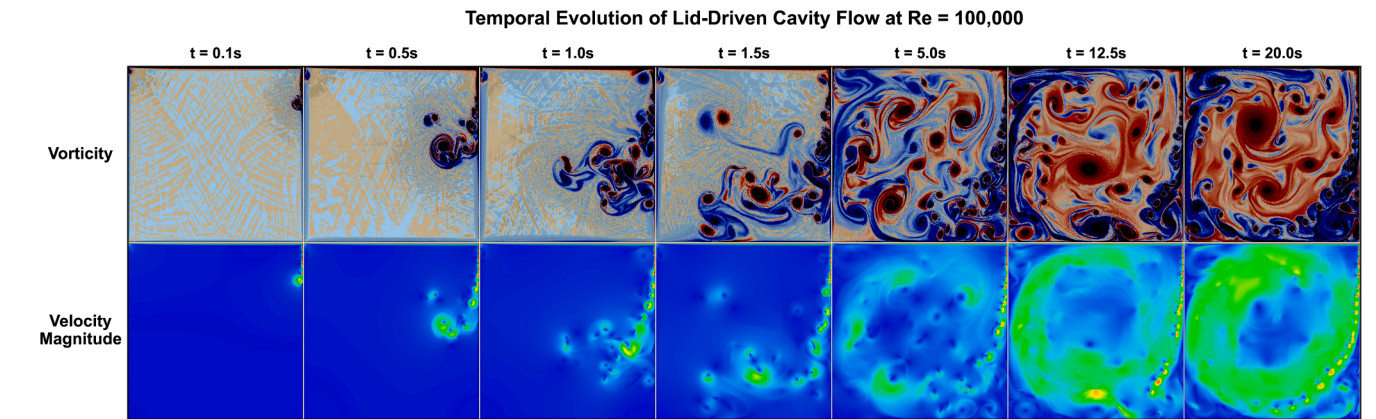
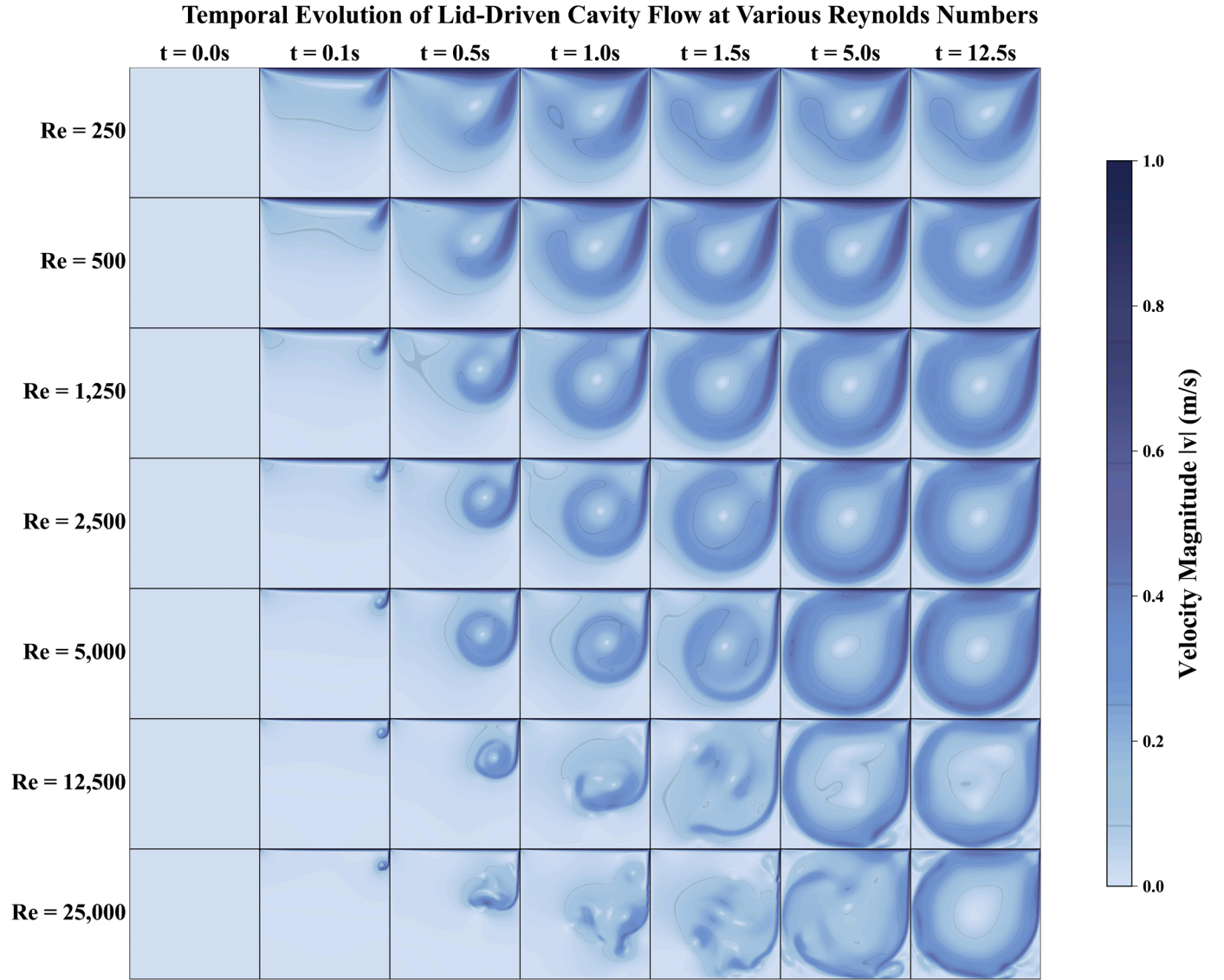


Fig. 13. Temporal evolution of vorticity contours for $Re = 250$ – 25000 at $t \in \{0.0, 0.1, 0.5, 1.0, 1.5, 5.0, 12.5\}$ s.



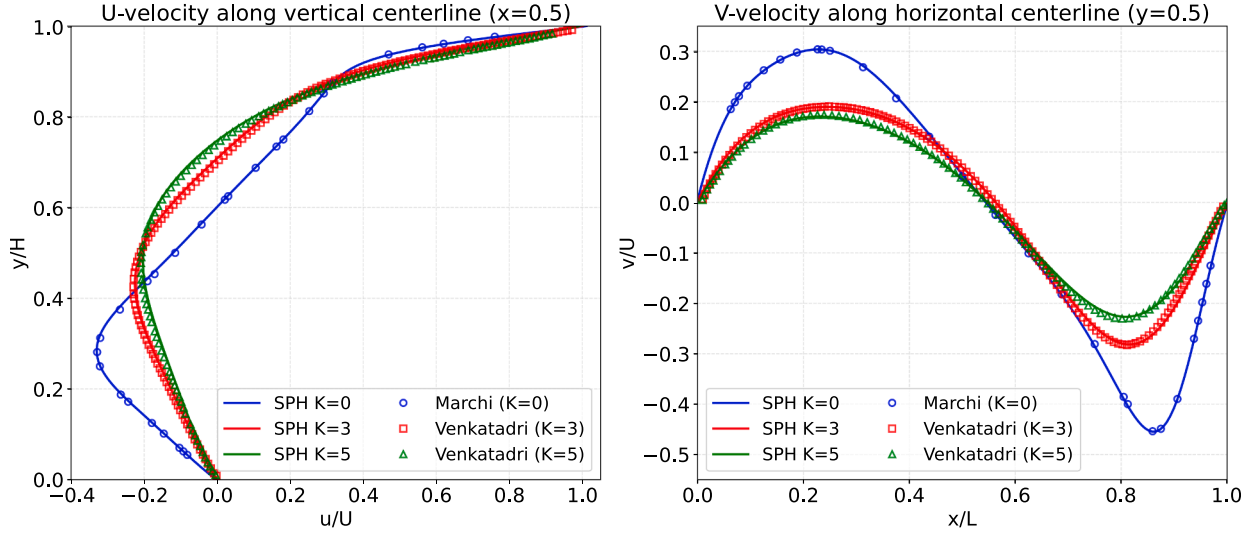


Fig. 16. Micropolar lid-driven cavity at $Re = 400$: velocity profiles for $K = 0, 3$, and 5 . (Left) u/U along vertical centerline; (Right) v/U along horizontal centerline. Present SPH results (solid lines) compared with Marchi et al. [44] ($K = 0$, circles) and Venkatadri et al. [45] ($K = 3$, squares; $K = 5$, triangles).

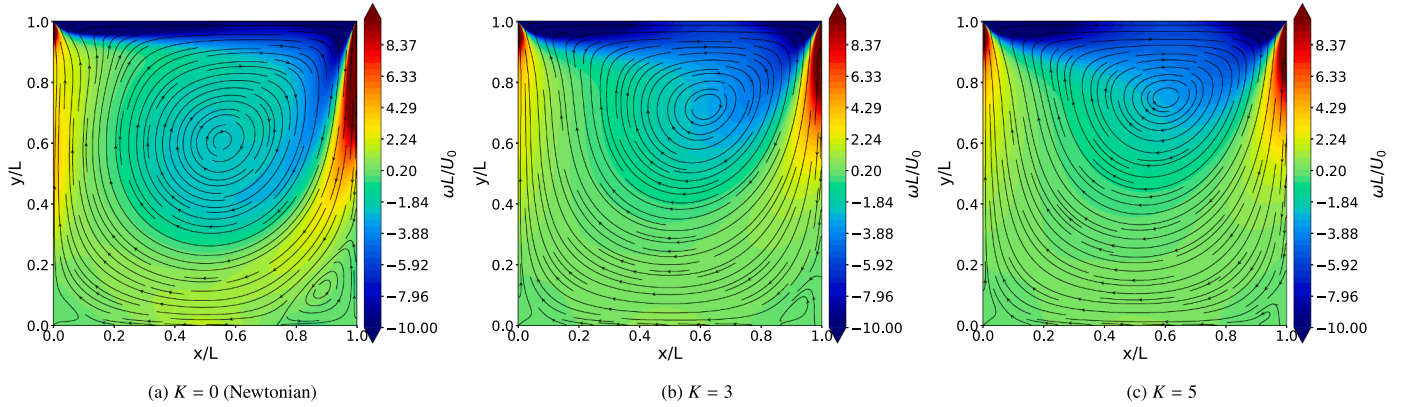


Fig. 17. Streamlines and vorticity ($\omega L/U_0$) for micropolar lid-driven cavity at $Re = 400$ with $K = 0, 3$, and 5 .

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + (\mu + \kappa) \nabla^2 \mathbf{v} + \kappa \nabla \times \mathbf{w}, \quad (53)$$

$$\rho j \frac{D\mathbf{w}}{Dt} = \gamma \nabla^2 \mathbf{w} + \kappa (\nabla \times \mathbf{v} - 2\mathbf{w}), \quad (54)$$

where μ is the dynamic viscosity, κ is the vortex viscosity, j is the micro-inertia density, and γ is the spin-gradient viscosity. The dimensionless vortex viscosity parameter $K = \kappa/\mu$ controls the micropolar effects; $K = 0$ recovers the Newtonian case.

We simulate the lid-driven cavity at $Re = 400$ with $K \in \{0, 3, 5\}$, enforcing no-spin boundary conditions ($\mathbf{w} = 0$) on all walls. Fig. 16 compares centerline velocity profiles with the benchmark data of Marchi et al. [44] for $K = 0$ and Venkatadri et al. [45] for $K = 3$ and 5 . The SPH results accurately capture the characteristic reduction in velocity magnitudes with increasing K .

Fig. 17 shows the streamline patterns and vorticity fields. As K increases, the primary vortex shifts upward toward the moving lid while secondary corner vortices are progressively suppressed. This behavior reflects the physical effect of increasing K : stronger resistance to relative rotation between fluid elements reduces overall flow intensity and modifies recirculation patterns.

The agreement with reference finite difference solutions validates the micropolar implementation and confirms the framework's capability for simulating fluids with microstructure.

Having validated the enhanced δ^+ -SPH formulation for confined viscous flows, we now extend our investigation to external flows, beginning with the elliptical cylinder before proceeding to NACA airfoil geometries.

4.2. Elliptical cylinder validation

4.2.1. Problem description

The elliptical cylinder serves as an intermediate test case between the lid-driven cavity and more complex aerodynamic geometries. This configuration retains geometric simplicity while introducing essential external flow physics: boundary layer development, flow separation, vortex shedding, and the drag crisis phenomenon. We validate against the viscous vortex domains (VVD) simulations of Dymnikova [46], who computed flow around an elliptical cylinder at Reynolds numbers up to $Re = 10^6$, along with experimental measurements [47–49].

We simulate two-dimensional flow around an elliptical cylinder with aspect ratio $a : b = 2 : 1$, where a and b are the semi-major and semi-minor axes, respectively. The cylinder is oriented with its major axis aligned with the freestream direction. The Reynolds number and drag coefficient are defined as:

$$Re = \frac{2aU}{\nu}, \quad C_d = \frac{F_d}{\frac{1}{2}\rho U^2(2a)}, \quad (55)$$

Table 5
Simulation parameters for the elliptical cylinder problem.

Parameter	Symbol	Value
Semi-major axis	a	0.5 m
Semi-minor axis	b	0.25 m
Freestream velocity	U	1.0 m/s
Number of fluid particles	N	8,000,000
Initial particle spacing	Δx	0.001 m
Smoothing length	h	0.005 m
CFL number	–	0.15
Reference sound speed	c_0	200 m/s
Tait equation exponent	γ	1.4
Reference density	ρ_0	1.0 kg/m ³

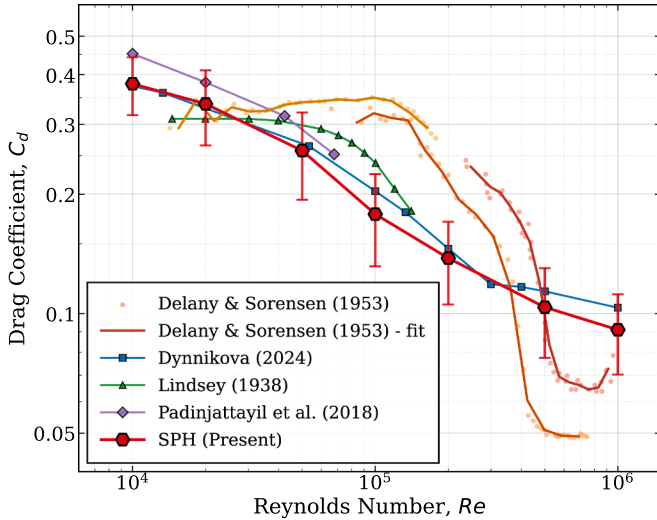


Fig. 18. Drag coefficient C_d versus Reynolds number for the elliptical cylinder ($a : b = 2 : 1$). Present SPH results (red hexagons) compared with VVD simulations [46] (blue squares) and experimental data [47–49]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where U is the freestream velocity, ν is the kinematic viscosity, and F_d is the drag force per unit span. We investigate Reynolds numbers spanning two orders of magnitude: $Re \in \{10^4, 2 \times 10^4, 5 \times 10^4, 10^5, 2 \times 10^5, 5 \times 10^5, 10^6\}$, encompassing laminar vortex shedding, turbulent transition, and the drag crisis regime.

4.2.2. Numerical configuration

The simulations employ the enhanced δ^+ -SPH formulation with GPU acceleration as described in Section 3. Table 5 summarizes the key parameters.

4.2.3. Results and discussion

Fig. 18 presents the drag coefficient as a function of Reynolds number. The SPH results show good agreement with the VVD benchmark of Dynnikova [46] across all regimes. In the laminar vortex shedding regime ($Re \leq 5 \times 10^4$), the drag coefficient decreases monotonically with increasing Reynolds number, following the well-established trend for bluff body flows. The relative differences in this regime remain within 4%, with excellent agreement at $Re = 10^4$ (1.4%) and $Re = 2 \times 10^4$ (2.6%). At intermediate Reynolds numbers ($10^5 \leq Re \leq 2 \times 10^5$), where the flow transitions from laminar to turbulent wake dynamics, the SPH results follow the same decreasing trend as the VVD data, with relative differences of 5–13%.

At high Reynolds numbers ($Re \geq 5 \times 10^5$), the simulation captures the drag crisis phenomenon, with C_d dropping to 0.104 ± 0.026 at $Re = 5 \times 10^5$ and 0.091 ± 0.021 at $Re = 10^6$. The relative differences with VVD

Table 6
Drag coefficient comparison between SPH and VVD [46].

Re	C_d (SPH)	$\pm\sigma$	C_d (VVD)	Diff. (%)
1×10^4	0.379	0.063	0.374	+1.4
2×10^4	0.338	0.072	0.329	+2.6
5×10^4	0.257	0.064	0.268	−4.0
1×10^5	0.178	0.047	0.203	−12.5
2×10^5	0.138	0.033	0.146	−5.4
5×10^5	0.104	0.026	0.114	−8.8
1×10^6	0.091	0.021	0.103	−12.0

predictions are 8.8% and 12.0%, respectively. It should be noted that the drag crisis is inherently three-dimensional, involving boundary layer transition and associated separation changes; two-dimensional simulations capture the qualitative trend but may not reproduce exact quantitative behavior observed in experiments.

Table 6 summarizes the quantitative comparison. At low Reynolds numbers ($Re \leq 5 \times 10^4$), the SPH results show excellent agreement with VVD data, with relative differences within 4%. At higher Reynolds numbers ($Re \geq 10^5$), the SPH results systematically under-predict drag by 5–13%, which is acceptable given the challenging transitional and drag crisis flow physics. The mean absolute difference across all Reynolds numbers is 6.7%.

Fig. 19 presents the instantaneous vorticity field at $tU_\infty/c = 10$ across the Reynolds number range, illustrating the flow physics underlying the drag coefficient trends. At $Re = 10^4$, the flow exhibits classical von Kármán vortex shedding with large, coherent alternating vortices forming the characteristic bluff-body wake. The boundary layer remains relatively thick and separates near the maximum thickness location, producing a wide wake region that corresponds to the high drag coefficient observed at this Reynolds number. As Reynolds number increases to $Re = 10^5$, the boundary layer thins and the separation point shifts downstream; the wake vortices become smaller but more numerous, indicating the onset of transitional behavior. At high Reynolds numbers ($Re \geq 5 \times 10^5$), the simulation captures the qualitative features of the drag crisis: the separation point moves further downstream toward the trailing edge, the wake narrows significantly, and fine-scale vortical structures appear on the cylinder surface. These flow features are directly responsible for the dramatic reduction in drag coefficient from $C_d \approx 0.38$ at $Re = 10^4$ to $C_d \approx 0.09$ at $Re = 10^6$.

Fig. 20 compares the temporal evolution of the vorticity field at two extreme Reynolds numbers. At $Re = 10^4$, the early-stage flow ($tU_\infty/c = 1$) shows symmetric vortex formation immediately behind the cylinder, which subsequently develops into a fully asymmetric von Kármán street by $tU_\infty/c = 10$. The vortices remain large and coherent throughout this evolution. In contrast, at $Re = 10^6$, the early-stage flow already exhibits fine-scale structures within the thin boundary layer, and the developed flow shows a narrow, turbulent wake with multiple small-scale vortices. The method's ability to resolve these multi-scale flow phenomena across two orders of magnitude in Reynolds number demonstrates its suitability for high-Reynolds-number external aerodynamic applications.

These results confirm the method's capability for external aerodynamic flows across two orders of magnitude in Reynolds number, providing confidence for the subsequent NACA airfoil simulations.

4.3. NACA airfoil validation

4.3.1. Problem description

The flow around NACA (National Advisory Committee for Aeronautics) airfoils represents a fundamental benchmark for aerodynamic simulations, providing critical validation for numerical methods in capturing complex phenomena including boundary layer development, flow separation, and wake dynamics. NACA profiles present unique challenges for particle-based methods due to their sharp leading and trailing edges, thin boundary layers at high Reynolds numbers, and the potential for

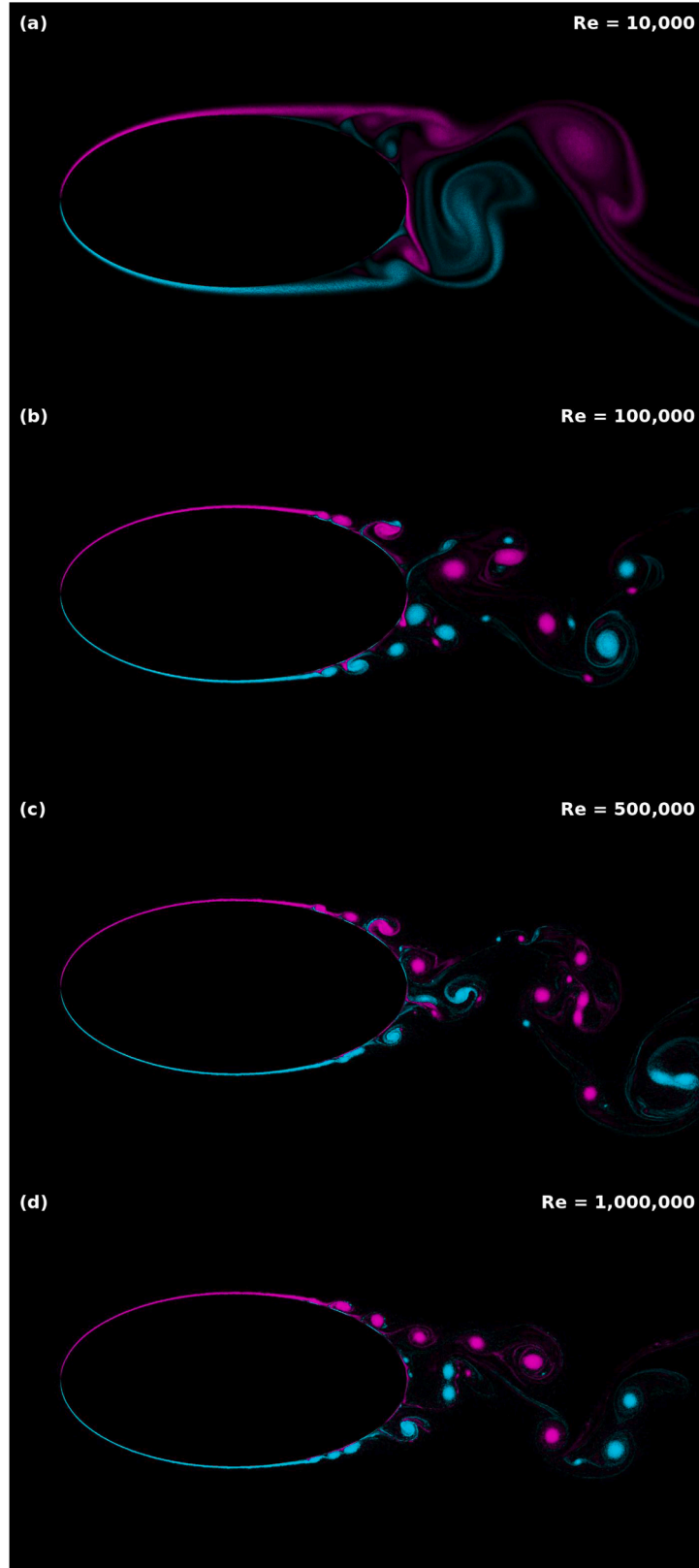


Fig. 19. Vorticity field around the elliptical cylinder ($a : b = 2 : 1$) at $tU_\infty/c = 10$ for Reynolds numbers spanning two orders of magnitude: (a) $Re = 10^4$, (b) $Re = 10^5$, (c) $Re = 5 \times 10^5$, and (d) $Re = 10^6$. Magenta and cyan indicate positive and negative vorticity, respectively. The progressive thinning of the boundary layer, downstream shift of the separation point, and narrowing of the wake with increasing Reynolds number illustrate the physical mechanisms underlying the drag crisis.

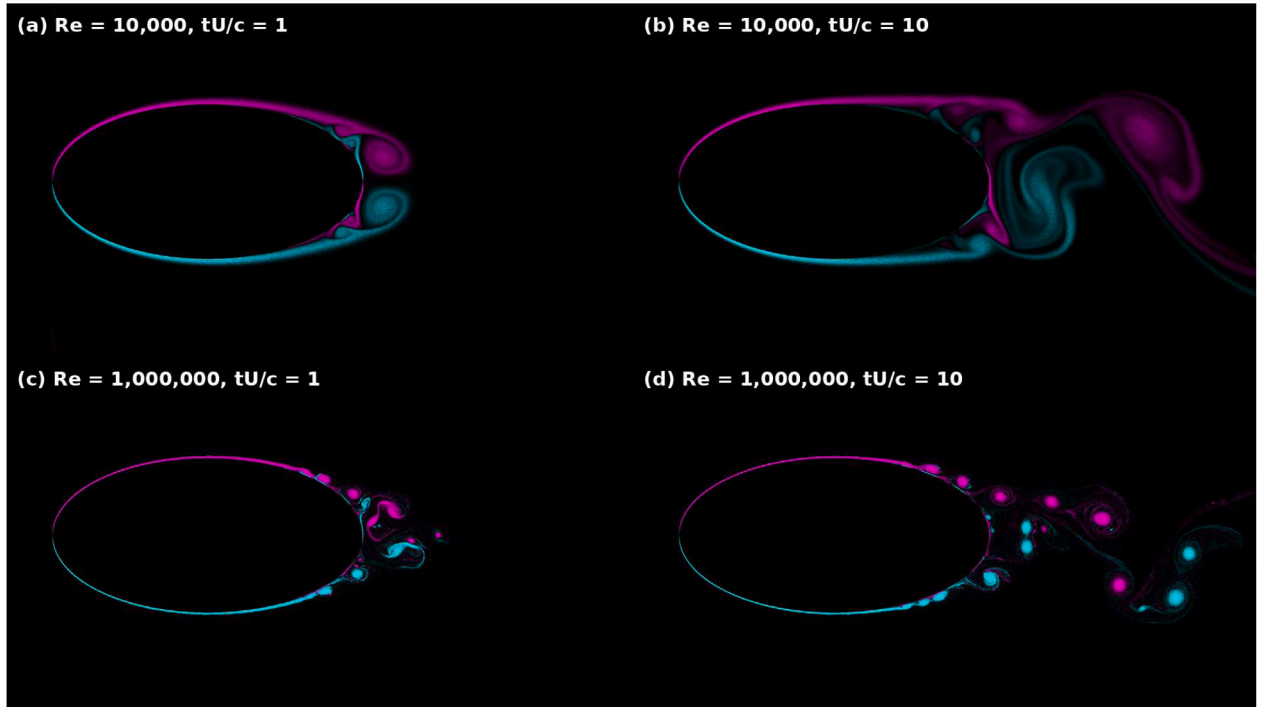


Fig. 20. Temporal evolution of the vorticity field around the elliptical cylinder comparing early ($tU_\infty/c = 1$, left column) and developed ($tU_\infty/c = 10$, right column) flow stages at $Re = 10^4$ (top row) and $Re = 10^6$ (bottom row): (a) $Re = 10^4$, $tU_\infty/c = 1$; (b) $Re = 10^4$, $tU_\infty/c = 10$; (c) $Re = 10^6$, $tU_\infty/c = 1$; (d) $Re = 10^6$, $tU_\infty/c = 10$. At low Reynolds number, the initial symmetric vortex pair evolves into a well-organized von Kármán street. At high Reynolds number, the thin boundary layer develops fine-scale instabilities that persist into the narrow wake, characteristic of the drag crisis regime.

strong adverse pressure gradients leading to flow separation. These features make NACA airfoil simulations an excellent test case to assess the robustness and accuracy of the micropolar δ^+ -SPH formulation, particularly its ability to handle regions of low or negative pressure without triggering tensile instability.

In this study, we investigate three distinct NACA profiles to comprehensively evaluate the performance of our method across multiple flow regimes. The selected airfoils, illustrated in Fig. 21, represent fundamentally different aerodynamic characteristics: NACA0008 is a thin symmetric airfoil with 8% maximum thickness, NACA0015 is a symmetric airfoil with 15% maximum thickness, and NACA6409 is a cambered airfoil featuring 6% maximum camber located at 40% chord and 9% maximum thickness.

The NACA0008 symmetric airfoil is tested under laminar to transitional flow conditions at moderate Reynolds numbers of $Re = 2,000$ and 6000 with an angle of attack $\alpha = 4^\circ$. These conditions allow us to examine the solver's capability in capturing unsteady vortex shedding and transitional flow phenomena typical of low-to-moderate Reynolds number flows.

The NACA0015 symmetric airfoil is examined at $Re = 180,000$ across a wide range of angles of attack $\alpha \in \{0^\circ, 5^\circ, 10^\circ, 15^\circ, 20^\circ, 23^\circ\}$. This configuration serves as a rigorous quantitative benchmark, as reference data are available from XFOIL viscous panel method calculations and recent two-dimensional diffused vortex hydrodynamics (DVH) simulations by Durante et al. [50], who investigated the turbulent transition of planar flow past the same airfoil at identical Reynolds number. The NACA0015 profile provides a well-documented test case spanning attached flow, stall onset, and post-stall regimes, enabling direct assessment of our solver's accuracy against established numerical benchmarks.

Finally, the NACA6409 cambered airfoil is examined across a wide range of Reynolds numbers ($Re = 12,500, 25,000$, and $200,000$) with angles of attack ranging from 0° to 20° , encompassing attached flow, moderate separation, and deep stall regimes. This configuration tests the solver's performance in capturing complex boundary layer behavior.

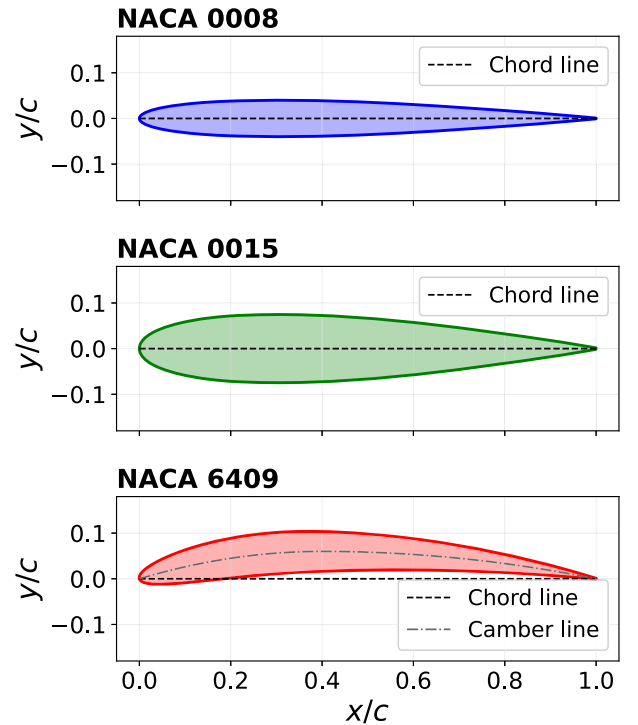


Fig. 21. Comparison of NACA0008, NACA0015, and NACA6409 airfoil profiles. The symmetric NACA0008 (blue) and NACA0015 (green) exhibit zero camber with 8% and 15% maximum thickness, respectively, while the NACA6409 (red) features significant camber with its maximum occurring at 40% of the chord length. All profiles are normalized by the chord length c . (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 7

Simulation parameters for the NACA0008/6409 airfoil cases.

Parameter	Symbol	Value
Chord length	c	0.1 m
Number of fluid particles	N	> 4,000,000
Freestream velocity	U_∞	1.0 m/s
Initial particle spacing	Δx	0.0002 m
Smoothing length	h	0.0008 m
CFL number	CFL	0.2
Reference sound speed	c_0	100 m/s
Tait equation exponent	γ	1.4

Table 8

Simulation parameters for the NACA0015 airfoil case.

Parameter	Symbol	Value
Chord length	c	1.0 m
Freestream velocity	U_∞	1.0 m/s
Reynolds number	Re_c	180,000
Angles of attack	α	0° – 23°
Number of fluid particles	N	$\approx 8,000,000$
Initial particle spacing	Δx	0.001 m
Smoothing length	h	0.005 m
CFL number	CFL	0.15
Reference sound speed	c_0	150 m/s
Reference density	ρ_0	1.0 kg/m ³
Tait equation exponent	γ	1.4

ior, flow separation characteristics, and the influence of the micropolar transfer coefficient on vorticity preservation at conditions relevant to practical aerodynamic applications.

The flow behavior is characterized by the Reynolds number based on the chord length:

$$Re_c = \frac{U_\infty c}{\nu}, \quad (56)$$

where U_∞ denotes the freestream velocity, c is the chord length, and ν is the kinematic viscosity of the fluid.

The aerodynamic performance is quantified through the drag coefficient C_d and lift coefficient C_l :

$$C_d = \frac{F_d}{\frac{1}{2} \rho U_\infty^2 c}, \quad C_l = \frac{F_l}{\frac{1}{2} \rho U_\infty^2 c}, \quad (57)$$

where F_d and F_l represent the drag and lift forces per unit span, respectively, computed using the pressure and viscous stress integration procedure proposed in [51].

4.3.2. Numerical configuration

The numerical simulations employ the micropolar δ^+ -SPH formulation with a weakly compressible approach based on the Tait equation of state. The computational domain extends from $-2c$ to $14c$ in the stream-wise direction and from $-2c$ to $2c$ in the transverse direction, with the airfoil center positioned at the origin. The simulation parameters for the NACA0008/6409 and NACA0015 cases are summarized in Tables 7 and 8, respectively.

The reference sound speed maintains a Mach number of approximately 0.01, ensuring that density variations remain below 1% consistent with the weakly compressible assumption. The airfoil surface is represented by fixed boundary particles with no-slip conditions. Simulations are initialized with the freestream velocity $U_\infty = 1.0$ m/s throughout the fluid domain. The adaptive timestep scheme based on the CFL criterion ensures numerical stability, while PST maintains uniform particle distribution in regions of high shear.

4.3.3. Results and discussion of NACA airfoil

Figs. 22 and 23 present the temporal evolution of the lift and drag coefficients for the NACA0008 airfoil at $Re_c = 2,000$ and 6,000 with $\alpha =$

4° . At $Re_c = 2,000$, the flow reaches a steady state after approximately 9 chord transit times, with the lift coefficient stabilizing at $C_l \approx 0.26$ and the drag coefficient at $C_d \approx 0.074$.

The present results demonstrate excellent agreement with the reference SPH data of Sun et al. [10] and the FVM and DVH results of Rossi et al. [52], validating our method's accuracy for laminar attached flow conditions. At $Re_c = 6,000$, the flow exhibits more complex behavior. After an initial transient period lasting approximately 3 chord transit times, the lift coefficient shows a characteristic overshoot reaching $C_l \approx 0.36$ before settling to a quasi-steady state with mild oscillations around $C_l \approx 0.245$. These oscillations, with amplitude $\Delta C_l \approx \pm 0.015$, indicate the onset of transitional flow behavior with periodic vortex shedding from the trailing edge. The drag coefficient similarly stabilizes around $C_d \approx 0.048$ with small periodic fluctuations. The excellent agreement with reference data validates our method's capability to capture transitional flow phenomena.

Having validated the solver at low Reynolds numbers with the thin NACA0008 profile, we now examine the NACA0015 airfoil at significantly higher Reynolds number $Re_c = 180,000$ across angles of attack from 0° to 23° to assess performance under more demanding flow conditions. Our simulation parameters are configured to match the two-dimensional diffused vortex hydrodynamics (DVH) study of Durante et al. [50], enabling direct comparison between our SPH implementation and their high-resolution Lagrangian vortex particle method. We also compare against XFOIL predictions and experimental data [53] reported in Durante et al. [50].

The NACA0015 profile presents a comprehensive validation challenge as the angle of attack range encompasses multiple flow regimes: attached flow with linear lift slope ($\alpha < 10^\circ$), stall onset with flow separation initiating from the trailing edge ($10^\circ < \alpha < 15^\circ$), and deep stall with massive separation from the leading edge ($\alpha > 15^\circ$). We present results from the classical Navier–Stokes formulation (corresponding to $\nu + \nu_t = 0$ in our micropolar framework) to enable direct comparison with the DVH simulations, which also solve the classical Navier–Stokes equations.

Fig. 24 presents the lift and drag coefficients compared with reference data. In the attached flow regime ($\alpha \leq 10^\circ$), the computed lift coefficients show excellent agreement with both XFOIL predictions and DVH simulations. The computed values at $\alpha = 5^\circ$ ($C_l = 0.52$) and $\alpha = 10^\circ$ ($C_l = 0.94$) agree with the DVH results ($C_l = 0.54$ and $C_l = 0.93$, respectively) within 3%, and the lift curve slope is consistent between both Lagrangian methods.

At higher angles of attack ($\alpha \geq 15^\circ$), the SPH results follow the DVH trend closely, both predicting continued lift increase beyond the stall angle predicted by XFOIL. This behavior is characteristic of two-dimensional simulations, where the absence of three-dimensional instabilities delays stall and permits higher maximum lift coefficients compared to experimental measurements. The computed $C_l = 1.29$ at $\alpha = 20^\circ$ agrees well with the DVH value of $C_l = 1.33$, demonstrating consistent behavior between the two Lagrangian particle methods. The drag coefficient predictions demonstrate similar trends, with good agreement at low angles where skin friction dominates. At higher angles ($\alpha = 20^\circ$ and 23°), the SPH predicts lower drag compared to DVH, likely due to insufficient resolution in the far-wake region. Table 9 provides a quantitative comparison, showing lift coefficient agreement within 7% across all angles.

To investigate the influence of micropolar effects on aerodynamic predictions, we repeat the NACA0015 simulations with the micropolar Eqs. (32) and (33) using $(\nu + \nu_t)/\nu = 5$. Fig. 25 presents the lift and drag coefficients obtained from the micropolar formulation compared with the same reference data.

The physical mechanism underlying the micropolar enhancement can be understood through the source term $(\nu + \nu_t)(\nabla \times \mathbf{v} - 2\mathbf{w})$ in the micro-rotation transport Eq. (33). This term measures the mismatch between the macroscopic vorticity $\nabla \times \mathbf{v}$ and the micro-rotation field $\mathbf{w}$. When the fluid vorticity begins to dissipate due to viscous effects or nu-

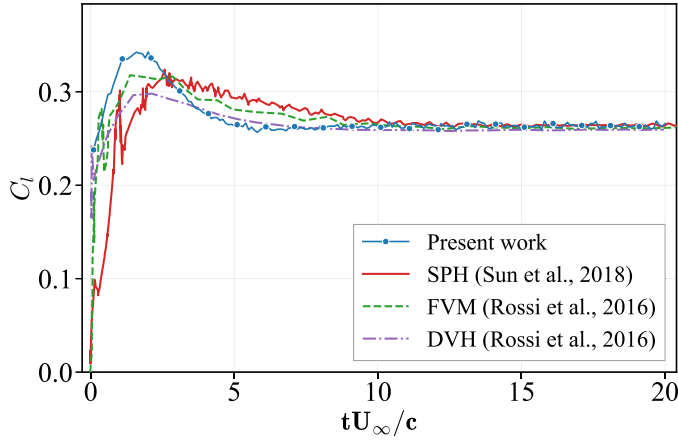
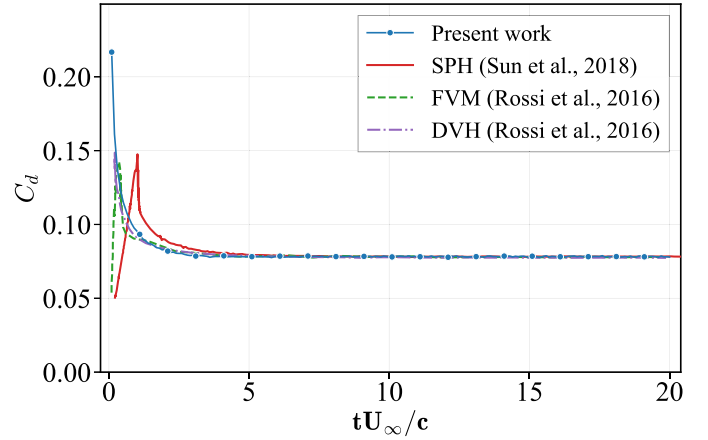
(a) NACA0008 Lift coefficient C_l at $Re_c = 2000$ with $\alpha = 4^\circ$ (b) NACA0008 Drag coefficient C_d at $Re_c = 2000$ with $\alpha = 4^\circ$

Fig. 22. Temporal evolution of aerodynamic coefficients for NACA0008 airfoil at $Re_c = 2,000$ and $\alpha = 4^\circ$. Present SPH results are compared with reference data from Sun et al. [10] (SPH), Rossi et al. [52] (FVM and DVH). The flow reaches a steady state after approximately 9 chord transit times, with the lift coefficient stabilizing at $C_l \approx 0.26$ and the drag coefficient at $C_d \approx 0.074$, demonstrating excellent agreement with reference solutions.

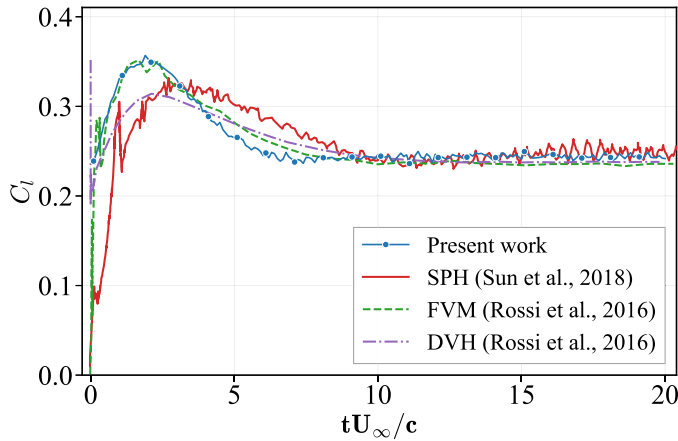
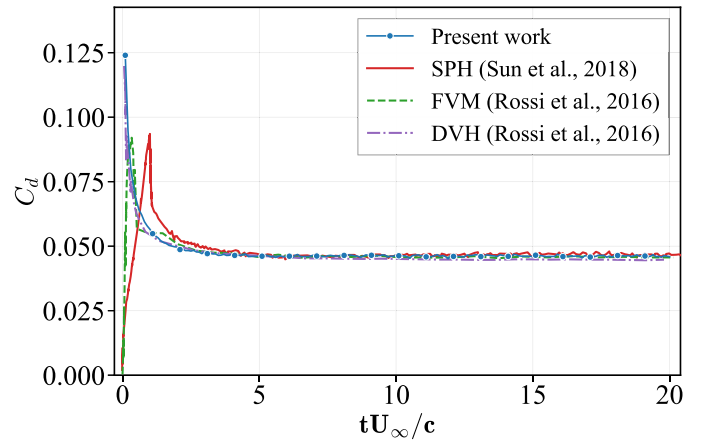
(a) NACA0008 Lift coefficient C_l at $Re_c = 6000$ with $\alpha = 4^\circ$ (b) NACA0008 Drag coefficient C_d at $Re_c = 6000$ with $\alpha = 4^\circ$

Fig. 23. Temporal evolution of aerodynamic coefficients for NACA0008 airfoil at $Re_c = 6,000$ and $\alpha = 4^\circ$. After an initial transient period of approximately 3 chord transit times, the lift coefficient exhibits a characteristic overshoot reaching $C_l \approx 0.36$ before settling to a quasi-steady state with mild oscillations around $C_l \approx 0.245$. The drag coefficient stabilizes around $C_d \approx 0.048$. These oscillations indicate the onset of transitional flow behavior with periodic vortex shedding. Reference data: Sun et al. [10] (SPH), Rossi et al. [52] (FVM and DVH).

Table 9

Comparison of aerodynamic coefficients for NACA0015 at $Re_c = 180,000$ between present SPH ($\nu + \nu_i = 0$) and DVH 2D [50].

α	C_l			C_d		
	SPH	DVH	Diff. (%)	SPH	DVH	Diff. (%)
0°	0.00	0.00	—	0.011	0.012	2.6
5°	0.52	0.54	2.2	0.017	0.016	3.6
10°	0.94	0.93	0.4	0.039	0.037	6.0
15°	1.21	1.30	6.5	0.113	0.110	2.6
20°	1.29	1.33	3.3	0.257	0.315	18.4
23°	1.24	1.20	2.8	0.362	0.516	29.8

merical diffusion, the micro-rotation field, which evolves according to its own transport equation, can feed angular momentum back into the velocity field through the coupling term $(\nu + \nu_i)\nabla \times \mathbf{w}$ in the momentum Eq. (32). This bidirectional coupling creates a vorticity preservation mechanism: angular momentum temporarily stored in the micro-rotation field resists rapid dissipation, allowing coherent vortical structures to persist longer than in classical Navier–Stokes simulations. In

separated flows, this mechanism stabilizes the shear layer roll-up and maintains organized vortex patterns on the airfoil upper surface, thereby reducing the chaotic breakdown and reformation of separation bubbles that would otherwise cause large temporal fluctuations in the aerodynamic forces.

In the attached flow regime ($\alpha \leq 10^\circ$), the micropolar formulation yields results essentially identical to the Navier–Stokes formulation, with lift and drag coefficients differing by less than 1%. This is expected because the flow remains attached and is dominated by pressure forces, with minimal vorticity generation in the wake. The micropolar coupling has negligible effect when rotational flow structures are weak.

At stall onset ($\alpha = 15^\circ$), the micropolar effects become significant. The micropolar formulation predicts $C_l = 1.28$, in close agreement with the DVH reference value ($C_l = 1.30$), compared to the Navier–Stokes prediction of $C_l = 1.21$. This improvement reflects the micropolar model's ability to maintain coherent vortical structures in the separating shear layer. Furthermore, the standard deviation of the lift coefficient decreases from $\sigma_{C_l} = 0.201$ (Navier–Stokes) to $\sigma_{C_l} = 0.157$ (micropolar), a reduction of 22%, indicating that the vorticity preservation mechanism

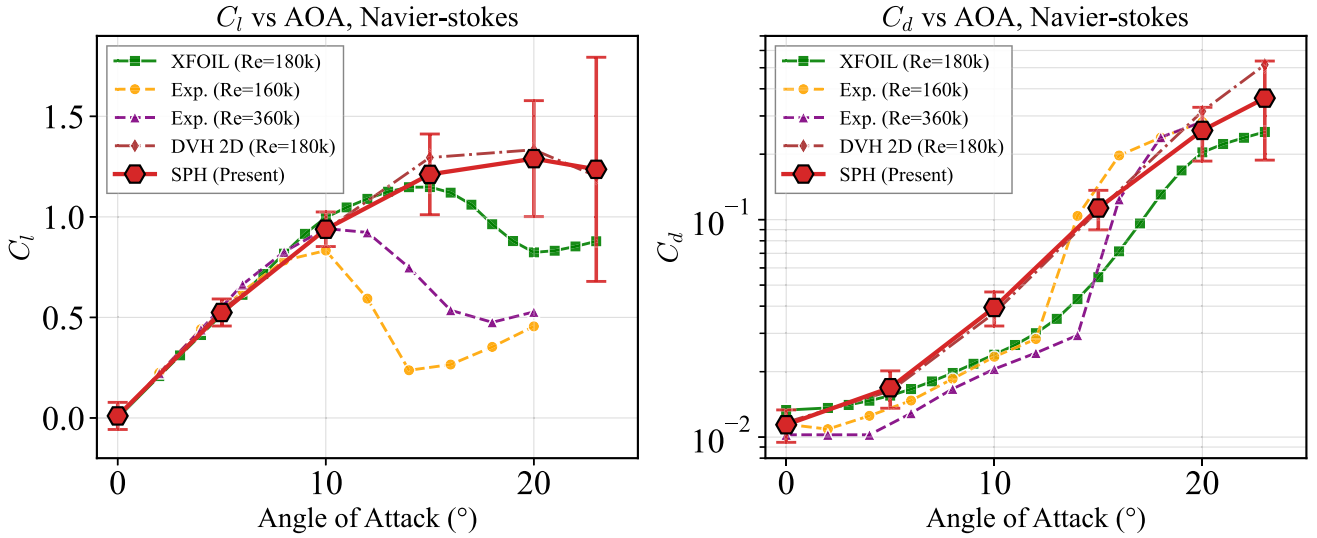


Fig. 24. Aerodynamic coefficients for NACA0015 airfoil at $Re_c = 180,000$ using the Navier–Stokes formulation ($\nu + \nu_t = 0$). Present SPH results (red hexagons with error bars) are compared with XFOIL calculations (green squares), experimental data at $Re_c = 160,000$ and $360,000$ [53] (orange circles and purple triangles), and DVH 2D simulations [50] (brown diamonds). Left: Lift coefficient C_l versus angle of attack. Right: Drag coefficient C_d versus angle of attack. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

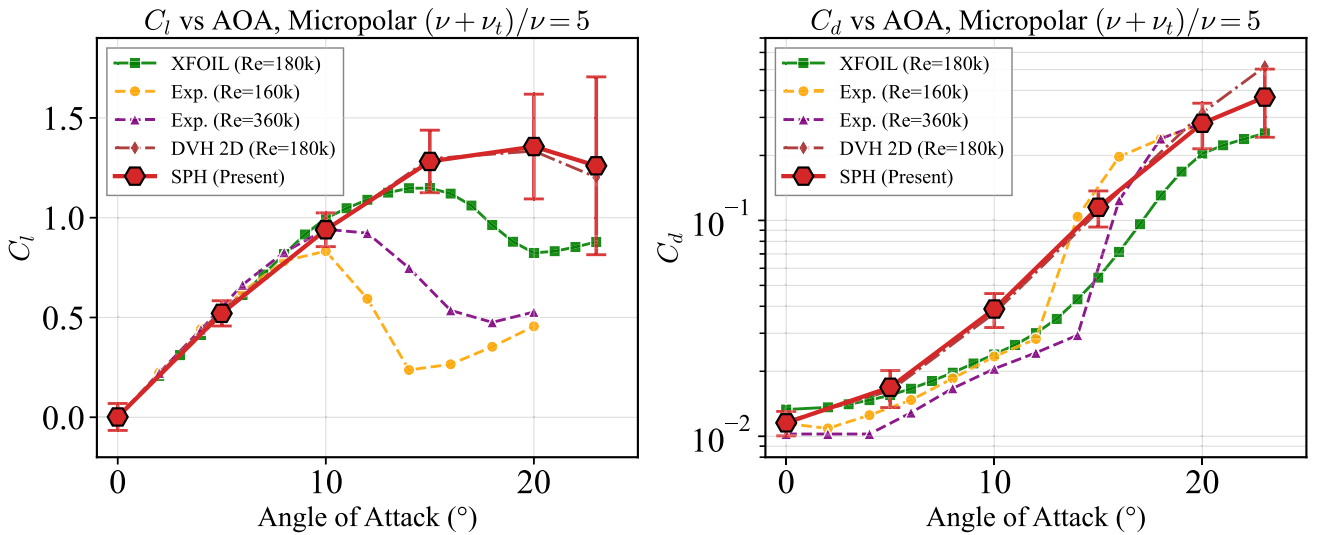


Fig. 25. Aerodynamic coefficients for NACA0015 airfoil at $Re_c = 180,000$ using the micropolar formulation ($(\nu + \nu_t)/\nu = 5$). Present SPH results (red hexagons with error bars) are compared with XFOIL calculations (green squares), experimental data at $Re_c = 160,000$ and $360,000$ [53] (orange circles and purple triangles), and DVH 2D simulations [50] (brown diamonds). Left: Lift coefficient C_l versus angle of attack. Right: Drag coefficient C_d versus angle of attack. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

reduces numerical dissipation and produces more temporally stable separated flow patterns.

In the post-stall regime ($\alpha = 20^\circ$ and 23°), the micropolar formulation continues to show improved agreement with DVH predictions. At $\alpha = 20^\circ$, the micropolar model predicts $C_l = 1.36$ and $C_d = 0.282$, compared to DVH values of $C_l = 1.33$ and $C_d = 0.315$. At $\alpha = 23^\circ$, the lift coefficient fluctuations decrease from $\sigma_{C_l} = 0.557$ to $\sigma_{C_l} = 0.445$ (20% reduction), demonstrating that the stabilizing effect of micropolar coupling persists in massively separated flows.

Fig. 26 directly compares the Navier–Stokes and micropolar formulations. The comparison confirms that micropolar effects are most pronounced in separated flow conditions when $\alpha \geq 15^\circ$, where the coupling between vorticity and micro-rotation helps stabilize vortical structures around the airfoil upper surface.

Table 10 summarizes the aerodynamic coefficients and their temporal fluctuations from both formulations. The micropolar formulation consistently improves lift coefficient predictions at high angles of attack while reducing temporal fluctuations, particularly at stall onset where the reduction in σ_{C_l} reaches 22%. These results demonstrate that the micropolar angular momentum transport mechanism provides meaningful improvements in both accuracy and stability for separated flow predictions.

We now turn to the cambered NACA6409 airfoil to examine the solver's capability in handling asymmetric profiles across a range of Reynolds numbers. Unlike the symmetric NACA profiles examined previously, the 6% camber of the NACA6409 generates lift even at zero geometric incidence, extending the validation to asymmetric pressure distributions characteristic of practical aerodynamic applications.

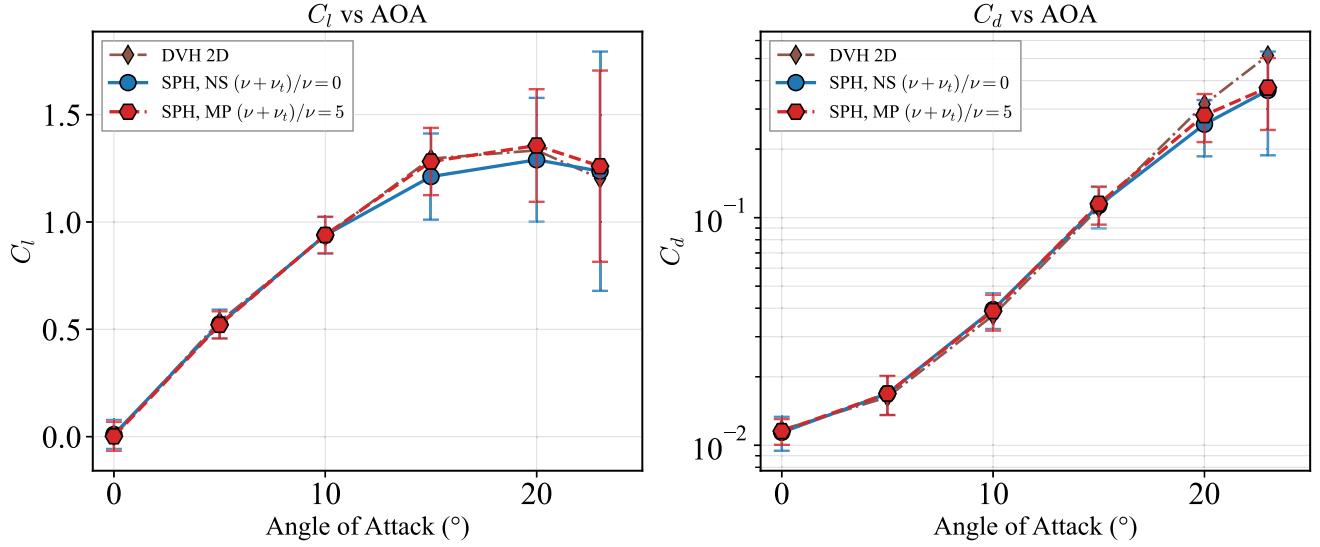


Fig. 26. Comparison of aerodynamic coefficients for NACA0015 airfoil at $Re_c = 180,000$ between Navier–Stokes $((\nu + \nu_t)/\nu = 0$, blue circles) and micropolar $((\nu + \nu_t)/\nu = 5$, red hexagons) formulations. DVH 2D simulations [50] (brown diamonds) are shown for reference. Left: Lift coefficient C_l versus angle of attack. Right: Drag coefficient C_d versus angle of attack. Error bars represent one standard deviation of temporal fluctuations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 10

Comparison of aerodynamic coefficients for NACA0015 at $Re_c = 180,000$ between Navier–Stokes (NS) and micropolar (MP) formulations. Standard deviations σ quantify temporal fluctuations.

α	Lift coefficient C_l				Drag coefficient C_d			
	NS	MP	σ_{NS}	σ_{MP}	NS	MP	σ_{NS}	σ_{MP}
0°	0.01	0.00	0.067	0.067	0.011	0.012	0.002	0.001
5°	0.52	0.52	0.067	0.063	0.017	0.017	0.003	0.003
10°	0.94	0.94	0.086	0.085	0.039	0.039	0.007	0.007
15°	1.21	1.28	0.201	0.157	0.113	0.115	0.023	0.022
20°	1.29	1.36	0.288	0.262	0.257	0.282	0.071	0.067
23°	1.24	1.26	0.557	0.445	0.362	0.373	0.174	0.130

Figs. 27 and 28 present the temporal evolution of aerodynamic coefficients at $Re_c = 200,000$ for $\alpha = 0^\circ$ and $\alpha = 8^\circ$, respectively. At $\alpha = 0^\circ$, the flow reaches steady state after approximately 4 chord transit times, with C_l stabilizing at 0.60 and C_d at 0.025. The computed lift coefficient shows excellent agreement with both experimental datasets: Selig et al. [54] report $C_l = 0.60$ and Prajapati et al. [55] measure $C_l = 0.594$. The rapid initial transient, characterized by negative lift at startup due to impulsive flow acceleration, demonstrates the method's robustness in handling strong pressure gradients.

At $\alpha = 8^\circ$, the airfoil operates in a high-lift regime with $C_l \approx 1.34$, closely matching the experimental values of 1.32 [54] and 1.29 [55]. The lift coefficient shows a sharp rise during the first two chord transit times, followed by stable convergence. The drag coefficient increases to $C_d \approx 0.053$, reflecting the increased pressure drag associated with mild trailing-edge separation at this higher incidence.

The lift coefficient predictions demonstrate excellent agreement with experimental data, within 2% at both angles of attack, consistent with the accuracy established in the NACA0008 and NACA0015 validations. However, the drag coefficient comparison requires careful interpretation. As demonstrated in the NACA0015 study (Fig. 24), direct comparison between 2D simulations and 3D wind tunnel measurements is inherently limited: both our SPH results and the DVH simulations of Durante et al. [50] showed excellent mutual agreement (within 7% for C_l) while

exhibiting systematic differences from experimental data, particularly at high angles of attack where three-dimensional stall mechanisms become dominant. This 2D-to-2D validation confirms the accuracy of our numerical implementation; discrepancies with 3D experimental data therefore reflect fundamental dimensional limitations rather than deficiencies in the SPH formulation.

For the NACA6409 drag predictions, the computed $C_d \approx 0.025$ at $\alpha = 0^\circ$ exceeds the value reported by Selig et al. [54] ($C_d = 0.0145$) but agrees well with Prajapati et al. [55] ($C_d = 0.023$). At $\alpha = 8^\circ$, the computed $C_d \approx 0.053$ exceeds both datasets: Selig et al. [54] report $C_d = 0.020$ and Prajapati et al. [55] measure $C_d = 0.035$. This over-prediction arises from two sources: (i) the experimental measurements include three-dimensional effects that reduce the effective profile drag contribution, and (ii) resolution limitations in capturing thin boundary-layer structures at high Reynolds numbers, where $\delta/c \sim Re_c^{-1/2} \approx 2 \times 10^{-3}$, a well-documented challenge for particle methods [10,51]. The considerable scatter between experimental datasets, with reported C_d values differing by up to 75% at $\alpha = 8^\circ$, further underscores the measurement challenges at this transitional Reynolds number regime.

To visualize the detailed flow physics and demonstrate the micropolar formulation's capability in capturing rotational structures, we present additional simulations at lower Reynolds numbers where the larger-scale vortical structures provide clearer illustration of the separation and wake dynamics.

Fig. 29 presents the microrotation field around the NACA6409 airfoil at $Re_c = 25,000$ for angles of attack from 0° to 20° . The micropolar formulation effectively captures rotational flow structures that would be significantly damped in classical SPH implementations. At low angles ($\alpha \leq 8^\circ$), the flow remains largely attached with microrotation concentrated in the wake region. The airfoil's camber generates asymmetric flow patterns even at zero incidence, while increasing angle progressively intensifies the wake vortex structures. At intermediate angles ($12^\circ \leq \alpha \leq 16^\circ$), flow separation initiates from the upper surface with the separation point advancing toward the leading edge; the microrotation field reveals recirculation regions and vortex shedding characteristic of partial stall. At $\alpha = 20^\circ$, full leading-edge separation produces chaotic vortical structures covering the entire upper surface. This successful

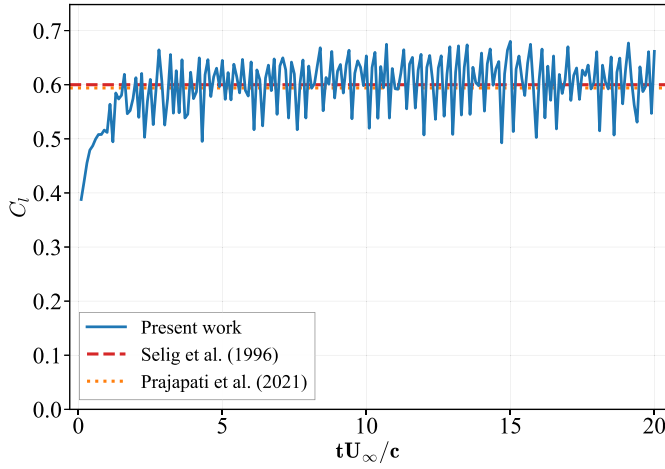
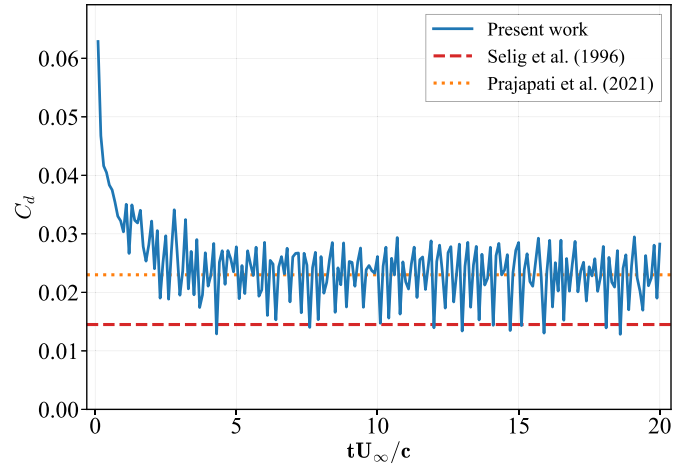
(a) NACA6409 Lift coefficient C_l at $Re_c = 200,000$ with $\alpha = 0^\circ$ (b) NACA6409 Drag coefficient C_d at $Re_c = 200,000$ with $\alpha = 0^\circ$

Fig. 27. Temporal evolution of aerodynamic coefficients for NACA6409 airfoil at $Re_c = 200,000$ and $\alpha = 0^\circ$. The airfoil's 6% camber generates lift even at zero geometric incidence. The flow reaches steady state after approximately 4 chord transit times, with C_l stabilizing at 0.60 and C_d at 0.025. Reference experimental data: Selig et al. [54] and Prajapati et al. [55].

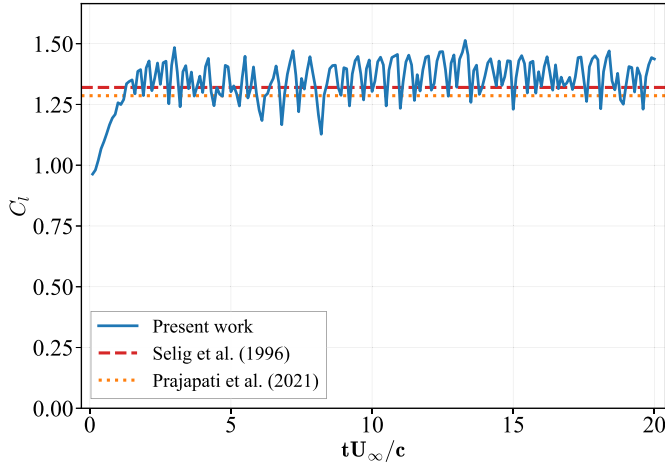
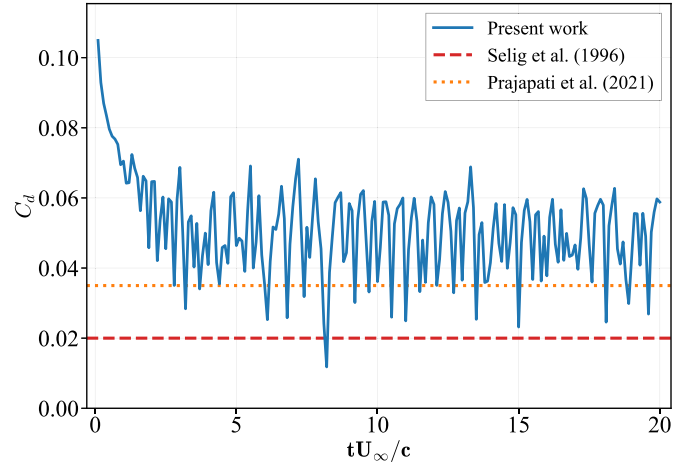
(a) NACA6409 Lift coefficient C_l at $Re_c = 200,000$ with $\alpha = 8^\circ$ (b) NACA6409 Drag coefficient C_d at $Re_c = 200,000$ with $\alpha = 8^\circ$

Fig. 28. Temporal evolution of aerodynamic coefficients for NACA6409 airfoil at $Re_c = 200,000$ and $\alpha = 8^\circ$. The airfoil operates in a high-lift regime with $C_l \approx 1.34$, in excellent agreement with experimental values of 1.32 [54] and 1.29 [55]. Reference data: Selig et al. [54] and Prajapati et al. [55].

capture of flow physics across all regimes demonstrates the effectiveness of the micropolar formulation in preserving rotational dynamics.

The influence of the micropolar transfer coefficient on vorticity preservation, previously quantified through aerodynamic coefficients in the NACA0015 study, can be directly visualized in the flow field.

Fig. 30 demonstrates the impact of varying $(\nu + \nu_t)$ on flow separation at $Re_c = 12,500$ and $\alpha = 20^\circ$, a condition featuring massive flow separation. Without micropolar effects $(\nu + \nu_t = 0)$, vortical structures rapidly dissipate due to numerical diffusion. With moderate coupling $(\nu + \nu_t = 0.005)$, coherent structures persist downstream. Under strong coupling $(\nu + \nu_t = 0.01)$, the separated shear layer rolls up into intense vortices that maintain their structure far into the wake, exhibiting energy cascade from large to small scales.

Fig. 31 provides further validation of the simulation's physical fidelity at $Re_c = 12,500$ and $\alpha = 10^\circ$. The vorticity field shows intense vortex generation at the trailing edge, where high-speed upper-surface flow meets low-speed lower-surface flow, consistent with the Kutta–

Joukowski condition requiring smooth flow departure. The wake exhibits alternating counter-rotating vortices with velocity minima at vortex cores, as expected from the Rankine vortex model. The velocity field between adjacent vortices shows the characteristic modulation predicted by the Biot–Savart law: constructive superposition between counter-rotating vortices produces high-velocity channels, while co-rotating regions exhibit velocity cancellation. This reproduction of fundamental vortex dynamics confirms the method's capability for aerodynamic applications.

5. Discussion

5.1. Framework performance and capabilities

The presented GPU-accelerated micropolar δ^+ -SPH framework offers four important advantages over conventional approaches, as demonstrated through the comprehensive validation studies.

First, the OpenGL compute-shader implementation achieves

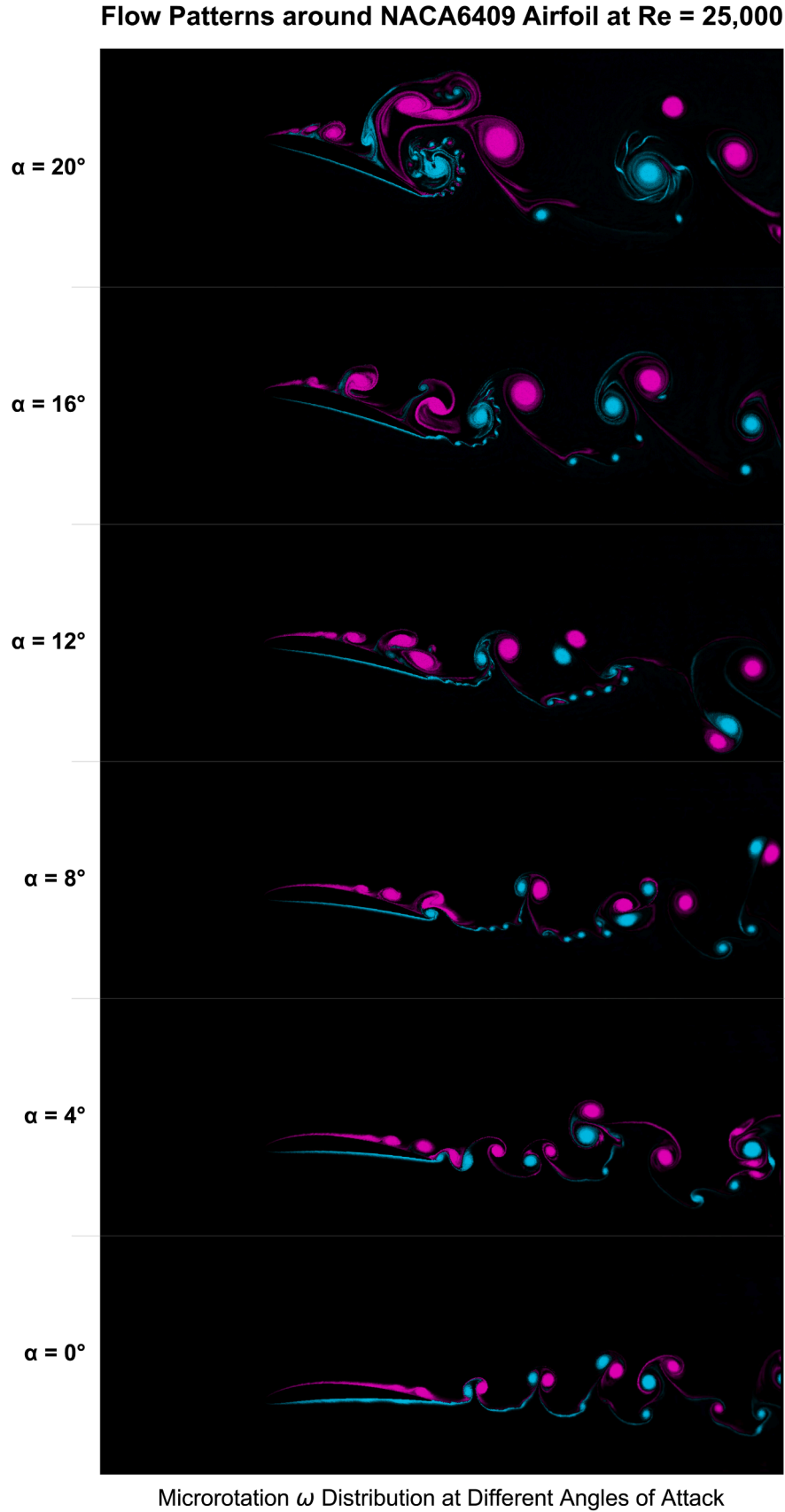
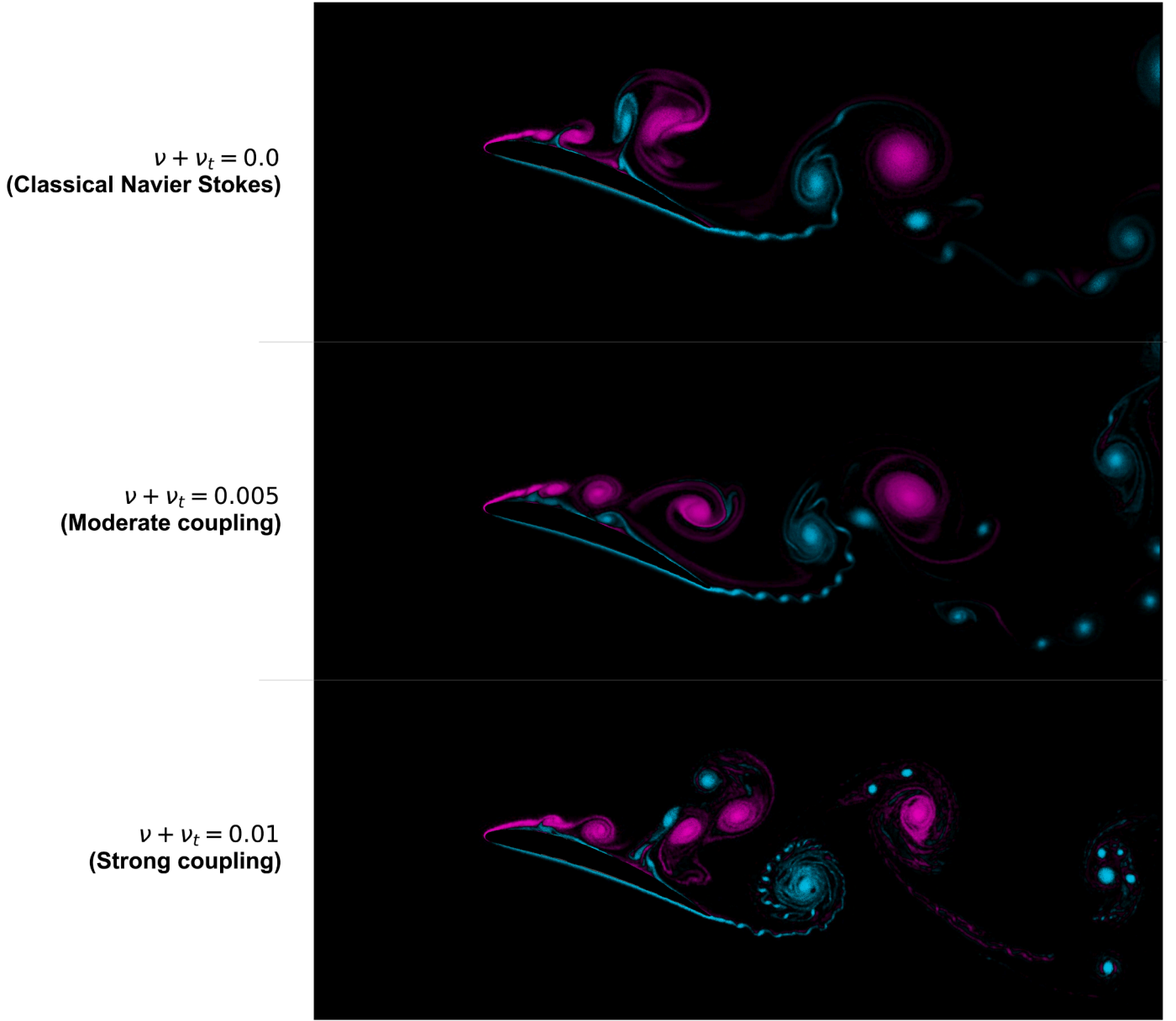


Fig. 29. Microrotation field ω distribution around NACA6409 airfoil at $Re_c = 25,000$ for angles of attack $\alpha \in \{0^\circ, 4^\circ, 8^\circ, 12^\circ, 16^\circ, 20^\circ\}$, demonstrating the micropolar SPH formulation's capability to capture rotational flow structures across attached, partially separated, and fully stalled flow regimes.

Effect of Micropolar Transfer Coefficient on Flow Separation



NACA6409 Airfoil at $Re = 12,500$, $\alpha = 20^\circ$

Fig. 30. Vorticity field $\nabla \times \mathbf{v}$ distribution around NACA6409 airfoil at $Re_c = 12,500$ and $\alpha = 20^\circ$ with varying micropolar transfer coefficients: (top) $\nu + \nu_t = 0$ (classical Navier–Stokes), (middle) $\nu + \nu_t = 0.005$ (moderate coupling), and (bottom) $\nu + \nu_t = 0.01$ (strong coupling). Increasing transfer coefficient enhances vorticity preservation and reveals progressively complex wake structures.

exceptional computational efficiency while maintaining portability across different GPU architectures. The demonstrated performance of more than 300 fps with one million particles on a single RTX 5090 achieves nearly two orders of magnitude improvement over optimized CPU implementations; peak throughput exceeds 400 million particles per second at 32 million particles (see Fig. 5). Direct benchmarking against DualSPHysics v5.4 reveals 17–18% reduction in wall-clock time for equivalent test cases, attributable to the efficient in-out buffer system for parallel particle migration and the periodic cache-optimized resorting strategy. This transforms SPH into a practical real-time simulation tool, enabling new interactive workflows in engineering and scientific applications.

Second, unlike vendor-specific solutions such as CUDA or ROCm, the OpenGL-based implementation provides seamless compatibility across multiple GPU vendors. Successful validation on both NVIDIA RTX 5090 and AMD RX 7900 XTX hardware demonstrates true vendor independence, achieving speedups of 37× and 41× over CPU baselines respectively. Compared to server-class hardware, the improvement is even more pronounced: the same 3D benchmark case requires 2339.6 s on an Intel Xeon Gold 6226R (16 cores) versus 24.1 s with the present work, representing a 97× speedup that approaches two orders of magnitude. Such cross-platform compatibility is particularly valuable in applied settings where heterogeneous computing resources are common, and lowers adoption barriers for both academic researchers and industrial users.

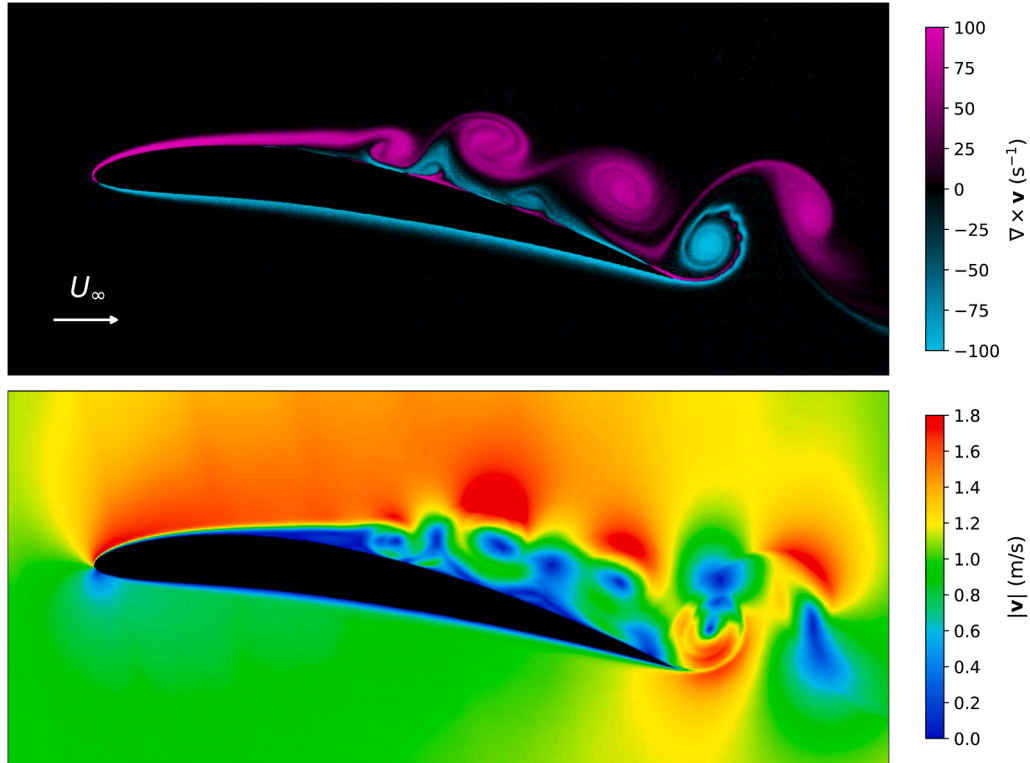


Fig. 31. Vorticity and velocity fields around NACA6409 airfoil at $Re_c = 12,500$ and $\alpha = 10^\circ$ with micropolar coefficient $\nu + \nu_i = 0.003 \text{ m}^2/\text{s}$. Top: Vorticity $\nabla \times \mathbf{v}$ showing enhanced preservation of rotational structures. Bottom: Velocity magnitude $|\mathbf{v}|$ revealing complex wake dynamics with well-maintained coherent vortical structures.

Third, the unified framework integrates multiple SPH formulations, allowing dynamic switching between Navier–Stokes and micropolar fluid models within a single computational pipeline. The quantitative validation studies demonstrate excellent accuracy across diverse flow regimes: lid-driven cavity simulations achieve relative L_2 errors below 1.3% compared to high-resolution finite difference benchmarks across Reynolds numbers from 100 to 10,000, while the micropolar formulation at $Re = 400$ accurately reproduces the velocity profile modifications induced by varying vortex viscosity parameters. The elliptical cylinder simulations capture the drag crisis phenomenon with mean differences of 6.7% relative to VVD reference data across two orders of magnitude in Reynolds number. NACA0015 airfoil simulations at $Re = 180,000$ show lift coefficient agreement within 7% of DVH reference solutions, with the micropolar formulation reducing temporal fluctuations by up to 22% compared to classical Navier–Stokes at stall conditions. This versatility enables systematic comparisons under identical conditions and supports the development of hybrid approaches.

Fourth, the numerical robustness of the framework is enhanced by the tensile instability control within the δ^+ -SPH scheme and by the improved particle shifting technique that maintains uniform particle distributions without explicit free-surface detection. The spatial convergence study confirms convergence rates of 1.3–1.6, consistent with theoretical expectations for SPH schemes with fixed $h/\Delta x$ ratios. These features suppress clustering and void formation, enabling stable simulations of flows with strong gradients, density variations, and complex boundaries.

Finally, the zero-copy architecture between computation and visualization eliminates traditional data-transfer bottlenecks, providing real-time parameter adjustment and immediate visual feedback. This integration transforms SPH from a batch-processing paradigm into an exploratory computational environment, significantly accelerating the de-

sign iteration cycle and opening opportunities for interactive engineering applications.

5.2. Limitations and future work

Despite its advantages, several limitations warrant consideration and suggest directions for future development.

The resolution requirements for accurate boundary layer capture at high Reynolds numbers remain computationally demanding. While lift coefficient predictions show excellent agreement with reference data (within 7% across all validated cases), drag coefficients exhibit systematic over-prediction compared to three-dimensional experiments and CFD. This discrepancy is an expected consequence of the two-dimensional formulation: the absence of three-dimensional vortex stretching and breakdown mechanisms leads to stronger, more persistent vortical structures that increase form drag. This dimensional limitation is well-documented in two-dimensional CFD and does not indicate deficiencies in the numerical scheme itself. Future work will incorporate multi-resolution strategies to extend the framework to large-scale problems where global flow behavior must be captured while maintaining high fidelity in localized regions of interest, such as near solid boundaries or in wake structures, without incurring the computational cost of uniform fine resolution throughout the entire domain. Hierarchical spatial data structures for adaptive-resolution SPH have been explored in various contexts, including octree-based approaches [56,57] and nested hierarchical structures for distributed-memory systems [58]. The current voxel-based architecture provides a natural foundation for such extension through hierarchical subdivision, where coarse voxels contain pre-computed mappings to finer voxels at multiple levels, enabling spatially varying smoothing lengths while preserving the locality and paralleliz-

ability that are central to the present GPU implementation. This hierarchical approach would maintain the $\mathcal{O}(N)$ computational complexity of the current implementation while enabling adaptive refinement in regions of high gradient or geometric complexity.

Current consumer GPU memory limitations restrict simulations to approximately 30–60 million particles, which may be insufficient for some industrial applications requiring extremely fine resolution or large computational domains. While professional GPUs offer larger memory capacities, their cost may limit accessibility. Future developments should explore multi-GPU parallelization and advanced memory management techniques to overcome these constraints, enabling billion-particle simulations on clustered GPU systems.

The micropolar formulation introduces additional parameters (ν_r , Θ , ξ) that must be calibrated for different flow regimes. While our validation demonstrates that appropriate parameter selection improves both accuracy and temporal stability in separated flows, the lack of universal selection guidelines currently requires case-specific validation. The observed 20–22% reduction in lift coefficient fluctuations with micropolar coupling at stall conditions suggests that the mechanism provides physically meaningful vorticity preservation, but systematic calibration studies across broader parameter spaces would enhance usability for non-expert users. Machine learning-based approaches for automatic parameter tuning represent a promising direction.

Beyond parameter calibration, the physical mechanism by which the micropolar formulation achieves improved vortex preservation merits discussion. In classical Navier–Stokes formulations, vorticity $\omega = \nabla \times \mathbf{v}$ is a derived quantity computed from velocity gradients, making it susceptible to numerical dissipation; any error in the velocity field immediately corrupts the vorticity field. The micropolar formulation addresses this limitation by introducing micro-rotation $\mathbf{w}$ as an independent kinematic variable carrying rotational information.

The key mechanism is the bidirectional coupling between velocity and micro-rotation fields. In the micro-rotation equation, the term $(\nu + \nu_r)(\nabla \times \mathbf{v} - 2\mathbf{w})$ represents the mismatch between local vorticity and micro-rotation; this mismatch generates a torque that drives micro-rotation to track the ambient vorticity field. Conversely, in the momentum equation, the term $(\nu + \nu_r)(\nabla \times \mathbf{w})$ provides feedback from micro-rotation gradients to the velocity field through couple stresses. This bidirectional coupling establishes a physically consistent mechanism where rotational information is stored redundantly in both the velocity gradient field and the independent micro-rotation field, providing inherent resistance to numerical dissipation of vortical structures.

The practical consequence of this mechanism is observed in the NACA airfoil simulations at stall conditions, where the micropolar formulation reduces temporal fluctuations in aerodynamic coefficients by 20–22% compared to classical Navier–Stokes (Table 10). This improvement reflects not merely numerical stabilization, but rather the more faithful representation of vortex dynamics enabled by the additional rotational degrees of freedom. The micropolar approach thus offers a physically motivated alternative to artificial numerical treatments commonly employed to suppress excessive vorticity dissipation in particle methods.

The comparison between two-dimensional simulations and three-dimensional experimental data presents inherent limitations, as demonstrated by both our SPH results and the DVH simulations of Durante et al., which show excellent mutual agreement while exhibiting systematic differences from wind tunnel measurements at high angles of attack. Future extensions should target three-dimensional simulations to capture the full physics of stall and wake dynamics.

Additional future extensions will target multiphase flows with varying density and viscosity, non-Newtonian fluids, thermal effects, and fluid–structure interactions. These developments will expand the framework's applicability to domains such as complex mixing processes, ship

hydrodynamics, and biomedical flow simulations, where the unique strengths of SPH are especially valuable.

6. Conclusions

This paper presented a GPU-accelerated micropolar δ^+ -SPH framework that addresses three fundamental challenges in particle-based CFD: computational cost, numerical fidelity, and real-time visualization.

The framework achieves exceptional computational performance, with benchmarks demonstrating 17–18% faster execution than DualSPHysics v5.4 on equivalent test cases and speedups approaching two orders of magnitude (97×) over server-class CPU implementations. The OpenGL compute shader architecture enables cross-platform execution on both NVIDIA and AMD GPUs without modification, achieving throughput exceeding 400 million particles per second on consumer hardware.

Comprehensive validation studies confirm the numerical accuracy of the implementation. Lid-driven cavity simulations achieve L_2 errors below 1.3% against high-resolution finite difference benchmarks across Reynolds numbers from 100 to 10,000, with spatial convergence rates of 1.3–1.6 consistent with theoretical expectations. External flow validations demonstrate mean drag coefficient differences of 6.7% for elliptical cylinders compared to VVD reference data, and lift coefficient agreement within 7% for NACA airfoils against DVH simulations. The micropolar formulation provides physically motivated vortex preservation through bidirectional coupling between velocity and micro-rotation fields, reducing aerodynamic coefficient fluctuations by 20–22% at stall conditions compared to classical Navier–Stokes.

The zero-copy architecture between computation and visualization eliminates traditional data-transfer bottlenecks, transforming SPH from a batch-processing tool into an interactive computational environment suitable for real-time engineering analysis.

In conclusion, this work demonstrates that modern SPH implementations can serve as viable alternatives to traditional grid-based methods for challenging fluid dynamics problems. The modular architecture supports future extensions to multiphase and non-Newtonian flows, as well as fluid–structure interaction, while real-time performance enables applications in interactive design and digital twin technologies. These advances open new possibilities in applied domains such as aerospace aerodynamics, marine hydrodynamics, and biomedical flow modeling, where large-scale, real-time CFD is increasingly in demand.

CRedit authorship contribution statement

Yixuan Chen: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Qiran Shao:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Yoshie Sugiyama:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

Yixuan Chen reports financial support was provided by Japan Science and Technology Agency. Qiran Shao reports financial support was provided by The University of Osaka and Daikin Industries, Ltd. Yoshie Sugiyama reports financial support was provided by Japan Science and

Technology Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank the anonymous reviewers and the editor for their constructive comments and valuable suggestions, which greatly helped improve the quality of this manuscript. The authors are grateful to Professor T. Yaguchi for his insightful advice. The authors are also grateful to Dr. Kosuke Shibata for his thoughtful comments and valuable suggestions. Y. Chen was supported by JST SPRING (Grant No.JPMJSP2138). Q. Shao was supported by the Student Research Program under the comprehensive collaboration agreement between The University of Osaka and Daikin Industries, Ltd. Y. Sugiyama was supported by JST CREST (Grant No. JPMJCR2013).

Appendix A. Lid-driven cavity validation data

This appendix provides detailed numerical data for the lid-driven cavity validation cases presented in Section 4.1. Tables A.1–A.17 contain the velocity profiles along cavity centerlines and error norm summaries for Reynolds numbers $Re \in \{100, 400, 1000, 3200, 7500, 10000\}$. The present SPH results are compared against the benchmark data of Ghia et al. [43] and the high-resolution finite difference solutions of Marchi et al. [44].

A.1. Reynolds number $Re = 100$

Tables A.1–A.3 present the velocity profiles and error norms at $Re = 100$.

Table A.1

Horizontal velocity u along the vertical centerline ($x = 0.5$) at $Re = 100$: Comparison with Ghia et al. [43] and Marchi et al. [44].

y	Present	Marchi	Ghia	Error ^a
1.0000	1.00000	–	1.00000	–
0.9766	0.84414	0.84348	0.84123	0.00066
0.9688	0.79274	0.79161	0.78871	0.00113
0.9609	0.74234	0.74071	0.73722	0.00163
0.9531	0.69317	0.69118	0.68717	0.00198
0.9375	0.60004	0.59747	–	0.00257
0.8750	0.31137	0.31056	–	0.00082
0.8516	0.23536	0.23644	0.23151	–0.00108
0.8125	0.13597	0.14043	–	–0.00445
0.7500	0.01986	0.02787	–	–0.00802
0.7344	–0.00419	0.00415	0.00332	–0.00834
0.6875	–0.06830	–0.06025	–	–0.00806
0.6250	–0.13653	–0.13126	–	–0.00528
0.6172	–0.14360	–0.13881	–0.13641	–0.00479
0.5625	–0.18312	–0.18208	–	–0.00104
0.5000	–0.20621	–0.20915	–0.20581	0.00294
0.4531	–0.20897	–0.21398	–0.21090	0.00501
0.4375	–0.20749	–0.21296	–	0.00548
0.3750	–0.19228	–0.19847	–	0.00619
0.3125	–0.16717	–0.17271	–	0.00554
0.2813	–0.15275	–0.15765	–0.15662	0.00490
0.2500	–0.13772	–0.14193	–	0.00421
0.1875	–0.10705	–0.10982	–	0.00276
0.1719	–0.09931	–0.10173	–0.10150	0.00242
0.1250	–0.07563	–0.07713	–	0.00150
0.1016	–0.06336	–0.06441	–0.06434	0.00105
0.0703	–0.04614	–0.04663	–0.04775	0.00049
0.0625	–0.04163	–0.04197	–0.04192	0.00035
0.0547	–0.03702	–0.03722	–0.03717	0.00020
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.2

Vertical velocity v along the horizontal centerline ($y = 0.5$) at $Re = 100$: Comparison with Ghia et al. [43] and Marchi et al. [44].

x	Present	Marchi	Ghia	Error ^a
1.0000	0.00000	–	0.00000	–
0.9688	–0.06496	–0.06229	–0.05906	–0.00267
0.9609	–0.08042	–0.07790	–0.07391	–0.00252
0.9531	–0.09566	–0.09334	–0.08864	–0.00232
0.9453	–0.11056	–0.10851	–0.10313	–0.00205
0.9375	–0.12509	–0.12332	–	–0.00177
0.9063	–0.17745	–0.17716	–0.16914	–0.00029
0.8750	–0.21748	–0.21869	–	0.00121
0.8594	–0.23188	–0.23371	–0.22445	0.00183
0.8125	–0.25094	–0.25377	–	0.00283
0.8047	–0.25068	–0.25354	–0.24533	0.00286
0.7500	–0.22558	–0.22783	–	0.00224
0.6875	–0.16232	–0.16301	–	0.00069
0.6250	–0.08480	–0.08407	–	–0.00073
0.5625	–0.00924	–0.00775	–	–0.00149
0.5000	0.05593	0.05754	0.05454	–0.00160
0.4375	0.10754	0.10878	–	–0.00124
0.3750	0.14517	0.14573	–	–0.00056
0.3125	0.16943	0.16913	–	0.00029
0.2500	0.18022	0.17924	–	0.00098
0.2344	0.18066	0.17956	0.17527	0.00110
0.2266	0.18050	0.17936	0.17507	0.00115
0.1875	0.17566	0.17434	–	0.00132
0.1563	0.16614	0.16480	0.16077	0.00134
0.1250	0.15057	0.14924	–	0.00133
0.0938	0.12767	0.12638	0.12317	0.00128
0.0781	0.11306	0.11178	0.10890	0.00128
0.0703	0.10491	0.10360	0.10091	0.00131
0.0625	0.09616	0.09481	0.09233	0.00135
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.3

Error norm summary at $Re = 100$: Comparison with Ghia et al. [43] and Marchi et al. [44].

Velocity	Reference	L1 Norm	L2 Norm	L_∞ Norm	Rel. L2 (%)
u (vertical center-line)	Ghia et al.	0.002825	0.003735	0.007510	0.82
	Marchi et al.	0.003316	0.004135	0.008340	1.21
v (horizontal center-line)	Ghia et al.	0.004825	0.005380	0.008310	4.03
	Marchi et al.	0.001488	0.001645	0.002860	1.05

A.2. Reynolds number $Re = 400$

Tables A.4–A.6 present the velocity profiles and error norms at $Re = 400$.

Table A.4

Horizontal velocity u along the vertical centerline ($x = 0.5$) at $Re = 400$: Comparison with Ghia et al. [43] and Marchi et al. [44].

y	Present	Marchi	Ghia	Error ^a
1.0000	1.00000	–	1.00000	–
0.9766	0.76160	0.76030	0.75837	0.00130
0.9688	0.68813	0.68676	0.68439	0.00138
0.9609	0.62170	0.62022	0.61756	0.00148
0.9531	0.56323	0.56174	0.55892	0.00149
0.9375	0.47045	0.46958	–	0.00087
0.8750	0.31447	0.31683	–	–0.00236
0.8516	0.28914	0.29200	0.29093	–0.00286
0.8125	0.24943	0.25220	–	–0.00278
0.7500	0.17911	0.18131	–	–0.00220
0.7344	0.16044	0.16254	0.16256	–0.00209
0.6875	0.10347	0.10546	–	–0.00198
0.6250	0.02793	0.03024	–	–0.00231
0.6172	0.01862	0.02102	0.02135	–0.00239
0.5625	–0.04577	–0.04257	–	–0.00320
0.5000	–0.11963	–0.11505	–0.11477	–0.00458
0.4531	–0.17709	–0.17145	–0.17119	–0.00564
0.4375	–0.19657	–0.19073	–	–0.00584
0.3750	–0.27126	–0.26631	–	–0.00496
0.3125	–0.32109	–0.32025	–	–0.00084
0.2813	–0.32699	–0.32872	–0.32726	0.00173
0.2500	–0.31749	–0.32123	–	0.00374
0.1875	–0.25897	–0.26392	–	0.00494
0.1719	–0.23913	–0.24375	–0.24299	0.00462
0.1250	–0.17538	–0.17875	–	0.00336
0.1016	–0.14384	–0.14614	–0.14612	0.00230
0.0703	–0.10232	–0.10332	–0.10338	0.00099
0.0625	–0.09189	–0.09260	–0.09266	0.00071
0.0547	–0.08137	–0.08180	–0.08186	0.00043
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.5

Vertical velocity v along the horizontal centerline ($y = 0.5$) at $Re = 400$: Comparison with Ghia et al. [43] and Marchi et al. [44].

x	Present	Marchi	Ghia	Error ^a
1.0000	0.00000	–	0.00000	–
0.9688	–0.12912	–0.12537	–0.12146	–0.00374
0.9609	–0.16469	–0.16141	–0.15663	–0.00328
0.9531	–0.20072	–0.19809	–0.19254	–0.00263
0.9453	–0.23666	–0.23466	–0.22847	–0.00200
0.9375	–0.27157	–0.27036	–	–0.00121
0.9063	–0.38825	–0.38983	–0.23827 ^b	0.00159
0.8750	–0.44701	–0.44901	–	0.00200
0.8594	–0.45244	–0.45383	–0.44993	0.00139
0.8125	–0.40160	–0.40004	–	–0.00155
0.8047	–0.38756	–0.38563	–0.38598	–0.00193
0.7500	–0.28349	–0.28099	–	–0.00250
0.6875	–0.18267	–0.18211	–	–0.00056
0.6250	–0.09984	–0.10088	–	0.00105
0.5625	–0.02244	–0.02471	–	0.00227
0.5000	0.05542	0.05206	0.05188	0.00336
0.4375	0.13454	0.13057	–	0.00397
0.3750	0.20986	0.20657	–	0.00329
0.3125	0.26954	0.26831	–	0.00123
0.2500	0.29986	0.30096	–	–0.00110
0.2344	0.30191	0.30345	0.30174	–0.00154
0.2266	0.30209	0.30383	0.30203	–0.00173
0.1875	0.29518	0.29748	–	–0.00230
0.1563	0.28126	0.28352	0.28124	–0.00225
0.1250	0.26030	0.26225	–	–0.00195
0.0938	0.23016	0.23166	0.22965	–0.00150
0.0781	0.20992	0.21100	0.20920	–0.00108
0.0703	0.19799	0.19881	0.19713	–0.00082
0.0625	0.18456	0.18513	0.18360	–0.00057
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

^b This value appears inconsistent with neighboring data points and may be a transcription error in the original reference.

Table A.6

Error norm summary at $Re = 400$: Comparison with Ghia et al. [43] and Marchi et al. [44].

Velocity	Reference	L1 Norm	L2 Norm	L_∞ Norm	Rel. L2 (%)
u (vertical center-line)	Ghia et al.	0.002444	0.003028	0.005900	0.71
	Marchi et al.	0.002621	0.003035	0.005840	0.93
v (horizontal center-line)	Ghia et al. [†]	0.002689	0.004194	0.008190	1.76
	Marchi et al.	0.001943	0.002148	0.003970	0.80

[†]The Ghia et al. data point at $x = 0.9063$ ($v = -0.23827$) was excluded from the error calculation as it appears to be a transcription error in the original reference.

A.3. Reynolds number $Re = 1000$

Tables A.7–A.9 present the velocity profiles and error norms at $Re = 1000$.

Table A.7

Horizontal velocity u along the vertical centerline ($x = 0.5$) at $Re = 1000$: Comparison with Ghia et al. [43] and Marchi et al. [44].

y	Present	Marchi	Ghia	Error ^a
1.0000	1.00000	–	1.00000	–
0.9766	0.65706	0.63973	0.65928	0.01733
0.9688	0.57528	0.58037	0.57492	-0.00508
0.9609	0.51309	0.51718	0.51117	-0.00409
0.9531	0.46830	0.47245	0.46604	-0.00415
0.9375	0.41817	0.42293	–	-0.00476
0.8750	0.35875	0.36255	–	-0.00380
0.8516	0.33419	0.33718	0.33304	-0.00299
0.8125	0.28662	0.28844	–	-0.00182
0.7500	0.20677	0.20791	–	-0.00114
0.7344	0.18752	0.18864	0.18719	-0.00113
0.6875	0.13243	0.13357	–	-0.00114
0.6250	0.06409	0.06525	–	-0.00116
0.6172	0.05584	0.05700	0.05702	-0.00117
0.5625	-0.00062	-0.00056	–	-0.00006
0.5000	-0.06324	-0.06206	-0.06080	-0.00118
0.4531	-0.10933	-0.10818	-0.10648	-0.00115
0.4375	-0.12455	-0.12341	–	-0.00114
0.3750	-0.18483	-0.18373	–	-0.00110
0.3125	-0.24732	-0.24569	–	-0.00163
0.2813	-0.28259	-0.28043	-0.27805	-0.00216
0.2500	-0.32135	-0.31895	–	-0.00240
0.1875	-0.38438	-0.38441	–	0.00003
0.1719	-0.38752	-0.38857	-0.38289	0.00105
0.1250	-0.34558	-0.34784	–	0.00226
0.1016	-0.29857	-0.30037	-0.29730	0.00180
0.0703	-0.22330	-0.22293	-0.22220	-0.00038
0.0625	-0.20341	-0.20233	-0.20196	-0.00108
0.0547	-0.18284	-0.18125	-0.18109	-0.00159
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.8

Vertical velocity v along the horizontal centerline ($y = 0.5$) at $Re = 1000$: Comparison with Ghia et al. [43] and Marchi et al. [44].

x	Present	Marchi	Ghia	Error ^a
1.0000	0.00000	–	0.00000	–
0.9688	-0.23514	-0.22834	-0.21388	-0.00680
0.9609	-0.29851	-0.29338	-0.27669	-0.00513
0.9531	-0.35886	-0.35513	-0.33714	-0.00373
0.9453	-0.41251	-0.41029	-0.39188	-0.00222
0.9375	-0.45713	-0.45615	–	-0.00098
0.9063	-0.52479	-0.52642	-0.51550	0.00163
0.8750	-0.46727	-0.46778	–	0.00050
0.8594	-0.42618	-0.42639	-0.42665	0.00021
0.8125	-0.33024	-0.33157	–	0.00132
0.8047	-0.31872	-0.32020	-0.31966	0.00148
0.7500	-0.25156	-0.25338	–	0.00182
0.6875	-0.18033	-0.18168	–	0.00135
0.6250	-0.10970	-0.11080	–	0.00110
0.5625	-0.04079	-0.04184	–	0.00105
0.5000	0.02679	0.02580	0.02526	0.00099
0.4375	0.09393	0.09297	–	0.00096
0.3750	0.16154	0.16056	–	0.00098
0.3125	0.23254	0.23127	–	0.00127
0.2500	0.30813	0.30710	–	0.00102
0.2344	0.32605	0.32539	0.32235	0.00066
0.2266	0.33446	0.33403	0.33075	0.00043
0.1875	0.36675	0.36785	–	-0.00110
0.1563	0.37488	0.37692	0.37095	-0.00204
0.1250	0.36234	0.36504	–	-0.00270
0.0938	0.33068	0.33298	0.32627	-0.00229
0.0781	0.30828	0.30995	0.30353	-0.00167
0.0703	0.29490	0.29629	0.29012	-0.00139
0.0625	0.27967	0.28071	0.27485	-0.00103
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.9

Error norm summary at $Re = 1000$: Comparison with Ghia et al. [43] and Marchi et al. [44].

Velocity	Reference	L1 Norm	L2 Norm	L_∞ Norm	Rel. L2 (%)
u (vertical center-line)	Ghia et al.	0.001732	0.002180	0.004630	0.53
	Marchi et al.	0.002455	0.003999	0.017330	1.27
v (horizontal center-line)	Ghia et al.	0.007515	0.010971	0.021820	3.52
	Marchi et al.	0.001711	0.002209	0.006800	0.70

A.4. Reynolds number $Re = 3200$

Tables A.10–A.12 present the velocity profiles and error norms at $Re = 3200$.

Table A.10

Horizontal velocity u along the vertical centerline ($x = 0.5$) at $Re = 3200$: Comparison with Ghia et al. [43] and Marchi et al. [44].

y	Present	Marchi	Ghia	Error ^a
1.0000	1.00000	–	1.00000	–
0.9766	0.52244	0.52772	0.53236	–0.00529
0.9688	0.47614	0.48099	0.48296	–0.00485
0.9609	0.46002	0.46542	0.46547	–0.00540
0.9531	0.45637	0.46180	0.46101	–0.00544
0.9375	0.45332	0.45740	–	–0.00408
0.8750	0.38327	0.38201	–	0.00126
0.8516	0.34997	0.34819	0.34682	0.00178
0.8125	0.29778	0.29580	–	0.00199
0.7500	0.22173	0.21974	–	0.00199
0.7344	0.20372	0.20182	0.19791	0.00191
0.6875	0.15168	0.15013	–	0.00155
0.6250	0.08594	0.08509	–	0.00085
0.6172	0.07796	0.07721	0.07156	0.00075
0.5625	0.02308	0.02314	–	–0.00006
0.5000	–0.03787	–0.03690	–0.04272	–0.00097
0.4531	–0.08282	–0.08122	–0.08664	–0.00161
0.4375	–0.09771	–0.09591	–	–0.00180
0.3750	–0.15717	–0.15453	–	–0.00264
0.3125	–0.21635	–0.21317	–	–0.00318
0.2813	–0.24600	–0.24262	–0.24427	–0.00339
0.2500	–0.27564	–0.27213	–	–0.00351
0.1875	–0.33392	–0.33036	–	–0.00356
0.1719	–0.34938	–0.34556	–0.34323	–0.00382
0.1250	–0.40961	–0.40508	–	–0.00453
0.1016	–0.43529	–0.43292	–0.41933	–0.00237
0.0703	–0.40853	–0.40832	–0.37827	–0.00021
0.0625	–0.38564	–0.38542	–0.35344	–0.00022
0.0547	–0.35696	–0.35651	–0.32407	–0.00046
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.11

Vertical velocity v along the horizontal centerline ($y = 0.5$) at $Re = 3200$: Comparison with Ghia et al. [43] and Marchi et al. [44].

x	Present	Marchi	Ghia	Error ^a
1.0000	0.00000	–	0.00000	–
0.9688	–0.44721	–0.43974	–0.39017	–0.00747
0.9609	–0.52279	–0.51979	–0.47425	–0.00301
0.9531	–0.56119	–0.56101	–0.52357	–0.00017
0.9453	–0.56574	–0.56724	–0.54053	0.00150
0.9375	–0.54812	–0.54962	–	0.00150
0.9062	–0.44219	–0.44351	–0.44307	0.00132
0.8750	–0.39449	–0.39627	–	0.00179
0.8594	–0.37746	–0.37875	–0.37401	0.00130
0.8125	–0.32294	–0.32310	–	0.00016
0.8047	–0.31363	–0.31368	–0.31184	0.00004
0.7500	–0.24998	–0.24961	–	–0.00037
0.6875	–0.18041	–0.18021	–	–0.00020
0.6250	–0.11365	–0.11371	–	0.00006
0.5625	–0.04844	–0.04918	–	0.00074
0.5000	0.01565	0.01426	0.00999	0.00139
0.4375	0.07942	0.07742	–	0.00200
0.3750	0.14346	0.14098	–	0.00248
0.3125	0.20830	0.20552	–	0.00278
0.2500	0.27407	0.27146	–	0.00261
0.2344	0.29063	0.28815	0.28188	0.00248
0.2266	0.29894	0.29654	0.29030	0.00240
0.1875	0.34172	0.33952	–	0.00219
0.1562	0.37912	0.37725	0.37119	0.00188
0.1250	0.41561	0.41553	–	0.00009
0.0938	0.42970	0.43275	0.42768	–0.00305
0.0781	0.41841	0.42246	0.41906	–0.00405
0.0703	0.40699	0.41129	0.40917	–0.00430
0.0625	0.39209	0.39615	0.39560	–0.00406
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.12

Error norm summary at $Re = 3200$: Comparison with Ghia et al. [43] and Marchi et al. [44].

Velocity	Reference	L1 Norm	L2 Norm	L_∞ Norm	Rel. L2 (%)
u (vertical center-line)	Ghia et al.	0.010003	0.014650	0.032890	3.58
	Marchi et al.	0.002479	0.002975	0.005430	0.93
v (horizontal center-line)	Ghia et al.	0.012581	0.021617	0.057040	5.80
	Marchi et al.	0.001977	0.002565	0.007470	0.71

A.5. Reynolds number $Re = 7500$

Tables A.13–A.15 present the velocity profiles and error norms at $Re = 7500$.

Table A.13

Horizontal velocity u along the vertical centerline ($x = 0.5$) at $Re = 7500$: Comparison with Ghia et al. [43] and Marchi et al. [44].

y	Present	Marchi	Ghia	Error ^a
1.0000	1.00000	–	1.00000	–
0.9766	0.48141	0.48858	0.47244	–0.00717
0.9688	0.48447	0.48920	0.47048	–0.00472
0.9609	0.48841	0.49252	0.47323	–0.00412
0.9531	0.48720	0.49078	0.47167	–0.00358
0.9375	0.47283	0.47501	–	–0.00218
0.8750	0.38185	0.38082	–	0.00104
0.8516	0.35006	0.34866	0.34228	0.00140
0.8125	0.29973	0.29817	–	0.00156
0.7500	0.22508	0.22395	–	0.00113
0.7344	0.20738	0.20642	0.20591	0.00096
0.6875	0.15623	0.15575	–	0.00048
0.6250	0.09168	0.09172	–	–0.00005
0.6172	0.08383	0.08394	0.08342	–0.00012
0.5625	0.03022	0.03051	–	–0.00029
0.5000	–0.02930	–0.02893	–0.03800	–0.00037
0.4531	–0.07334	–0.07282	–0.07503	–0.00052
0.4375	–0.08800	–0.08737	–	–0.00063
0.3750	–0.14662	–0.14534	–	–0.00128
0.3125	–0.20538	–0.20314	–	–0.00224
0.2813	–0.23478	–0.23203	–0.23176	–0.00274
0.2500	–0.26409	–0.26094	–	–0.00315
0.1875	–0.32275	–0.31889	–	–0.00387
0.1719	–0.33732	–0.33345	–0.32393	–0.00387
0.1250	–0.37981	–0.37639	–	–0.00342
0.1016	–0.40263	–0.40015	–0.38324	–0.00249
0.0703	–0.44831	–0.44808	–0.43025	–0.00023
0.0625	–0.45553	–0.45477	–0.43590	–0.00076
0.0547	–0.45381	–0.45130	–0.43154	–0.00251
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.14

Vertical velocity v along the horizontal centerline ($y = 0.5$) at $Re = 7500$: Comparison with Ghia et al. [43] and Marchi et al. [44].

x	Present	Marchi	Ghia	Error ^a
1.0000	0.00000	–	0.00000	–
0.9688	–0.57828	–0.57069	–0.53858	–0.00759
0.9609	–0.57140	–0.57441	–0.55216	0.00301
0.9531	–0.53230	–0.53827	–0.52347	0.00598
0.9453	–0.49407	–0.49916	–0.48590	0.00509
0.9375	–0.46951	–0.47194	–	0.00244
0.9063	–0.43015	–0.42941	–0.41050	–0.00074
0.8750	–0.39303	–0.39221	–	–0.00082
0.8594	–0.37363	–0.37295	–0.36213	–0.00068
0.8125	–0.31765	–0.31717	–	–0.00048
0.8047	–0.30847	–0.30813	–0.30448	–0.00034
0.7500	–0.24604	–0.24654	–	0.00050
0.6875	–0.17810	–0.17936	–	0.00126
0.6250	–0.11310	–0.11478	–	0.00168
0.5625	–0.05038	–0.05196	–	0.00158
0.5000	0.01095	0.00989	0.00824	0.00106
0.4375	0.07210	0.07145	–	0.00065
0.3750	0.13406	0.13333	–	0.00074
0.3125	0.19715	0.19596	–	0.00120
0.2500	0.26165	0.25976	–	0.00188
0.2344	0.27799	0.27596	0.27348	0.00204
0.2266	0.28619	0.28409	0.28117	0.00210
0.1875	0.32756	0.32531	–	0.00225
0.1563	0.36094	0.35895	0.35060	0.00199
0.1250	0.39472	0.39359	–	0.00113
0.0938	0.43337	0.43361	0.41824	–0.00024
0.0781	0.45176	0.45245	0.43564	–0.00069
0.0703	0.45729	0.45767	0.44030	–0.00038
0.0625	0.45799	0.45741	0.43979	0.00058
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.15

Error norm summary at $Re = 7500$: Comparison with Ghia et al. [43] and Marchi et al. [44].

Velocity	Reference	L1 Norm	L2 Norm	L_∞ Norm	Rel. L2 (%)
u (vertical center-line)	Ghia et al.	0.009969	0.012511	0.022270	3.01
	Marchi et al.	0.002030	0.002645	0.007170	0.81
v (horizontal center-line)	Ghia et al.	0.011771	0.015134	0.039700	3.93
	Marchi et al.	0.001753	0.002466	0.007590	0.68

A.6. Reynolds number $Re = 10000$

Tables A.16–A.18 present the velocity profiles and error norms at $Re = 10000$.

Table A.16

Horizontal velocity u along the vertical centerline ($x = 0.5$) at $Re = 10000$: Comparison with Ghia et al. [43] and Marchi et al. [44].

y	Present	Marchi	Ghia	Error ^a
1.0000	1.00000	–	1.00000	–
0.9766	0.49317	0.49249	0.47221	0.00069
0.9688	0.49825	0.49928	0.47783	–0.00103
0.9609	0.49771	0.50091	0.48070	–0.00319
0.9531	0.49123	0.49605	0.47804	–0.00482
0.9375	0.47038	0.47530	–	–0.00492
0.8750	0.37958	0.38049	–	–0.00091
0.8516	0.34840	0.34858	0.34635	–0.00018
0.8125	0.29889	0.29849	–	0.00040
0.7500	0.22528	0.22481	–	0.00047
0.7344	0.20770	0.20738	0.20673	0.00031
0.6875	0.15680	0.15698	–	–0.00018
0.6250	0.09262	0.09320	–	–0.00057
0.6172	0.08486	0.08544	0.08344	–0.00058
0.5625	0.03169	0.03214	–	–0.00045
0.5000	–0.02730	–0.02720	–0.03111	–0.00010
0.4531	–0.07106	–0.07102	–0.07540	–0.00004
0.4375	–0.08561	–0.08554	–	–0.00007
0.3750	–0.14403	–0.14339	–	–0.00064
0.3125	–0.20272	–0.20103	–	–0.00169
0.2813	–0.23214	–0.22981	–0.23186	–0.00233
0.2500	–0.26155	–0.25857	–	–0.00299
0.1875	–0.31900	–0.31606	–	–0.00293
0.1719	–0.33299	–0.33047	–0.32709	–0.00252
0.1250	–0.37478	–0.37368	–	–0.00111
0.1016	–0.39589	–0.39468	–0.38000	–0.00121
0.0703	–0.43813	–0.43691	–0.41657	–0.00122
0.0625	–0.45190	–0.45043	–0.42537	–0.00146
0.0547	–0.46101	–0.45867	–0.42735	–0.00233
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.17

Vertical velocity v along the horizontal centerline ($y = 0.5$) at $Re = 10000$: Comparison with Ghia et al. [43] and Marchi et al. [44].

x	Present	Marchi	Ghia	Error ^a
1.0000	0.00000	–	0.00000	–
0.9688	–0.58274	–0.58219	–0.54302	–0.00056
0.9609	–0.54503	–0.55313	–0.52987	0.00811
0.9531	–0.50337	–0.50840	–0.49099	0.00503
0.9453	–0.47616	–0.47796	–0.45863	0.00180
0.9375	–0.46157	–0.46208	–	0.00052
0.9063	–0.42688	–0.42803	–0.41496	0.00115
0.8750	–0.38933	–0.38959	–	0.00025
0.8594	–0.37143	–0.37064	–0.36737	–0.00079
0.8125	–0.31780	–0.31563	–	–0.00217
0.8047	–0.30897	–0.30669	–0.30719	–0.00228
0.7500	–0.24715	–0.24573	–	–0.00142
0.6875	–0.17909	–0.17914	–	0.00005
0.6250	–0.11395	–0.11507	–	0.00112
0.5625	–0.05107	–0.05269	–	0.00162
0.5000	0.01028	0.00876	0.00831	0.00152
0.4375	0.07133	0.06994	–	0.00139
0.3750	0.13289	0.13140	–	0.00149
0.3125	0.19552	0.19356	–	0.00196
0.2500	0.25909	0.25681	–	0.00228
0.2344	0.27510	0.27285	0.27224	0.00225
0.2266	0.28310	0.28091	0.28003	0.00220
0.1875	0.32355	0.32169	–	0.00186
0.1563	0.35681	0.35508	0.35070	0.00173
0.1250	0.39099	0.38918	–	0.00181
0.0938	0.42818	0.42628	0.41487	0.00190
0.0781	0.44915	0.44788	0.43124	0.00128
0.0703	0.45845	0.45764	0.43733	0.00081
0.0625	0.46458	0.46397	0.43983	0.00060
0.0000	0.00000	–	0.00000	–

^a Error computed against Marchi et al. where available.

Table A.18

Error norm summary at $Re = 10000$: Comparison with Ghia et al. [43] and Marchi et al. [44].

Velocity	Reference	L1 Norm	L2 Norm	L_∞ Norm	Rel. L2 (%)
u (vertical center-line)	Ghia et al.	0.011058	0.015242	0.033660	3.67
	Marchi et al.	0.001406	0.001944	0.004920	0.59
v (horizontal center-line)	Ghia et al.	0.011391	0.015426	0.039720	4.07
	Marchi et al.	0.001783	0.002344	0.008100	0.65

References

- [1] L.B. Lucy, A numerical approach to the testing of the fission hypothesis, *Astron. J.* 82 (1977) 1013–1024. <https://doi.org/10.1086/112164>
- [2] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, *Mon. Not. R. Astron. Soc.* 181 (3) (1977) 375–389. <https://doi.org/10.1093/mnras/181.3.375>
- [3] M.B. Liu, G.R. Liu, Smoothed particle hydrodynamics (SPH): an overview and recent developments, *Arch. Comput. Methods Eng.* 17 (1) (2010) 25–76. <https://doi.org/10.1007/s11831-010-9040-7>
- [4] J.J. Monaghan, Smoothed particle hydrodynamics and its diverse applications, *Annu. Rev. Fluid Mech.* 44 (2012) 323–346. <https://doi.org/10.1146/annurev-fluid-120710-101220>
- [5] D. Nishitani, M. Furuichi, H. Sakaguchi, Computational performance of a smoothed particle hydrodynamics simulation for shared-memory parallel computing, *Comput. Phys. Commun.* 194 (2015) 18–32. <https://doi.org/10.1016/j.cpc.2015.04.006>
- [6] D. Winkler, M. Rezavand, W. Rauch, Neighbour lists for smoothed particle hydrodynamics on GPUs, *Comput. Phys. Commun.* 225 (2018) 140–148. <https://doi.org/10.1016/j.cpc.2017.12.014>
- [7] A.J.C. Crespo, J.M. Domínguez, A. Barreiro, M. Gómez-Gesteira, B.D. Rogers, GPUs, a new tool of acceleration in CFD: efficiency and reliability on smoothed particle hydrodynamics methods, *PLoS ONE* 6 (6) (2011) e20685. <https://doi.org/10.1371/journal.pone.0020685>
- [8] R. Vacondio, C. Altomare, M. De Lefte, X. Hu, D. Le Touzé, S. Lind, J.-C. Marongiu, S. Marrone, B.D. Rogers, A. Souto-Iglesias, Grand challenges for smoothed particle hydrodynamics numerical schemes, *Comput. Part. Mech.* 8 (3) (2021) 575–588. <https://doi.org/10.1007/s40571-020-00354-1>
- [9] J. Bender, D. Koscier, T. Kugelstadt, M. Weiler, Turbulent micropolar SPH fluids with foam, *IEEE Trans. Vis. Comput. Graph.* 25 (6) (2019) 2284–2295. <https://doi.org/10.1109/TVCG.2018.2832080>
- [10] P.N. Sun, A. Colagrossi, S. Marrone, M. Antuono, A.M. Zhang, Multi-resolution delta-plus-SPH with tensile instability control: towards high Reynolds number flows, *Comput. Phys. Commun.* 224 (2018) 63–80. <https://doi.org/10.1016/j.cpc.2017.11.016>
- [11] L. Westhofen, S. Jeske, J. Bender, A comparison of linear consistent correction methods for first-order SPH derivatives, *Proc. ACM Comput. Graph. Interact. Tech.* 6 (3) (2023) 1–20. <https://doi.org/10.1145/3606933>
- [12] J.M. Domínguez, G. Fourtakas, C. Altomare, R.B. Canelas, A. Tafuni, O. García-Feal, I. Martínez-Estévez, A. Mokos, R. Vacondio, A.J.C. Crespo, B.D. Rogers, P.K. Stansby, M. Gómez-Gesteira, DualSPHysics: from fluid dynamics to multiphysics problems, *Comput. Part. Mech.* 9 (5) (2021) 867–895. <https://doi.org/10.1007/s40571-021-00404-2>
- [13] S. Band, C. Gissler, M. Teschner, Compressed neighbour lists for SPH, *Comput. Graph. Forum* 39 (1) (2020) 531–542. <https://doi.org/10.1111/cgf.13890>
- [14] J.M. Domínguez, A.J.C. Crespo, D. Valdez-Balderas, B.D. Rogers, M. Gómez-Gesteira, New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters, *Comput. Phys. Commun.* 184 (8) (2013) 1848–1860. <https://doi.org/10.1016/j.cpc.2013.03.008>
- [15] X. Xia, Q. Liang, A GPU-accelerated smoothed particle hydrodynamics (SPH) model for the shallow water equations, *Environ. Model. Softw.* 75 (2016) 28–43. <https://doi.org/10.1016/j.envsoft.2015.10.002>
- [16] D. Morikawa, H. Senadheera, M. Asai, Explicit incompressible smoothed particle hydrodynamics in a multi-GPU environment for large-scale simulations, *Comput. Part. Mech.* 8 (3) (2021) 493–510. <https://doi.org/10.1007/s40571-020-00347-0>
- [17] Y. Cai, J. Wei, Q. Hou, R. Gao, An optimized GPU implementation of weakly-compressible SPH using CUDA-based strategies, in: *Algorithms and Architectures for Parallel Processing (ICA3PP 2021)*, 13155 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 354–369. https://doi.org/10.1007/978-3-030-95384-3_23
- [18] Y. Cai, J. Wei, J. Duan, Q. Hou, An improved GPU acceleration framework for smoothed particle hydrodynamics, in: *Algorithms and Architectures for Parallel Processing (ICA3PP 2023)*, 14492 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 190–201. https://doi.org/10.1007/978-981-97-0811-6_11
- [19] Z. Mao, X. Li, S. Hu, G. Gopalakrishnan, A. Li, A GPU accelerated mixed-precision smoothed particle hydrodynamics framework with cell-based relative coordinates, *Eng. Anal. Bound. Elem.* 161 (2024) 113–125. <https://doi.org/10.1016/j.enganabound.2024.01.020>
- [20] A.J.C. Crespo, J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S.M. Longshaw, R.B. Canelas, R. Vacondio, A. Barreiro, O. García-Feal, DualSPHysics: open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH), *Comput. Phys. Commun.* 187 (2015) 204–216. <https://doi.org/10.1016/j.cpc.2014.10.004>
- [21] J. Bender, et al., SPLisHSPH Library, 2022, Accessed: 2025, <https://github.com/InteractiveComputerGraphics/SPLisHSPH>
- [22] J. Ahrens, S. Jourdain, P. O’Leary, J. Patchett, D.H. Rogers, M. Petersen, An image-based approach to extreme scale in situ visualization and analysis, in: *Proc. SC’14 Int. Conf. High Perform. Comput. Netw. Storage Anal., IEEE*, 2014, pp. 424–434. <https://doi.org/10.1109/SC.2014.40>
- [23] S. Adami, X.Y. Hu, N.A. Adams, A generalized wall boundary condition for smoothed particle hydrodynamics, *J. Comput. Phys.* 231 (21) (2012) 7057–7075. <https://doi.org/10.1016/j.jcp.2012.05.005>
- [24] M. Ferrand, D.R. Laurence, B.D. Rogers, D. Violeau, C. Kassiotis, Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method, *Int. J. Numer. Methods Fluids* 71 (4) (2013) 446–472. <https://doi.org/10.1002/fld.3666>
- [25] D. Le Touzé, A. Colagrossi, Smoothed particle hydrodynamics for free-surface and multiphase flows: a review, *Rep. Prog. Phys.* 88 (3) (2025) 037001. <https://doi.org/10.1088/1361-6633/ada80f>
- [26] G.R. Liu, M.B. Liu, *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*, World Scientific, Singapore, 2003.
- [27] J.J. Monaghan, Smoothed particle hydrodynamics, *Rep. Prog. Phys.* 68 (8) (2005) 1703–1759. <https://doi.org/10.1088/0034-4885/68/8/R01>
- [28] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, *Adv. Comput. Math.* 4 (1) (1995) 389–396. <https://doi.org/10.1007/BF02123482>
- [29] W. Dehnen, H. Aly, Improving convergence in smoothed particle hydrodynamics simulations without pairing instability, *Mon. Not. R. Astron. Soc.* 425 (2) (2012) 1068–1082. <https://doi.org/10.1111/j.1365-2966.2012.21439.x>
- [30] T.S. Tricco, The Kelvin-Helmholtz instability and smoothed particle hydrodynamics, *J. Phys. Conf. Ser.* 1225 (1) (2019) 012019. <https://doi.org/10.1088/1742-6596/1225/1/012019>
- [31] J.J. Monaghan, Smoothed particle hydrodynamics, *Annu. Rev. Astron. Astrophys.* 30 (1992) 543–574. <https://doi.org/10.1146/annurev.aa.30.090192.002551>
- [32] J. Bonet, T.S.L. Lok, Variational and momentum preservation aspects of smooth particle hydrodynamic formulations, *Comput. Methods Appl. Mech. Eng.* 180 (1) (1999) 97–115. [https://doi.org/10.1016/S0045-7825\(99\)00051-1](https://doi.org/10.1016/S0045-7825(99)00051-1)
- [33] J.J. Monaghan, Simulating free surface flows with SPH, *J. Comput. Phys.* 110 (2) (1994) 399–406. <https://doi.org/10.1006/jcp.1994.1034>
- [34] A.C. Eringen, Theory of micropolar fluids, *J. Math. Mech.* 16 (1) (1966) 1–18.
- [35] T. Ariman, On the analysis of blood flow, *J. Biomech.* 4 (3) (1971) 185–192. [https://doi.org/10.1016/0021-9290\(71\)90003-0](https://doi.org/10.1016/0021-9290(71)90003-0)
- [36] G. Lukaszewicz, *Micropolar Fluids: Theory and Applications*, Birkhäuser, Boston, MA, Boston, MA, 1999.
- [37] A. Souto-Iglesias, J. Bonet Avalos, M. Antuono, A. Colagrossi, General isotropic micropolar fluid model in smoothed particle hydrodynamics, *Phys. Rev. E* 104 (1) (2021) 015315. <https://doi.org/10.1103/PhysRevE.104.015315>
- [38] J. Chen, C. Liang, J.D. Lee, Theory and simulation of micropolar fluid dynamics, *Proc. Inst. Mech. Eng. N J. Nanoeng. Nanosyst.* 224 (1–2) (2010) 31–39. <https://doi.org/10.1177/1740349911400132>
- [39] S. Marrone, A. Colagrossi, D. Le Touzé, G. Graziani, Fast free-surface detection and level-set function definition in SPH solvers, *J. Comput. Phys.* 229 (10) (2010) 3652–3663. <https://doi.org/10.1016/j.jcp.2010.01.019>
- [40] S.J. Lind, R. Xu, P.K. Stansby, B.D. Rogers, Incompressible smoothed particle hydrodynamics for free-surface flows: a generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves, *J. Comput. Phys.* 231 (4) (2012) 1499–1523. <https://doi.org/10.1016/j.jcp.2011.10.027>
- [41] G. Bilotta, A. Hérault, A. Cappello, G. Ganci, C. Del Negro, GPUSPH: a smoothed particle hydrodynamics model for the thermal and rheological evolution of lava flows, in: *Detecting, Modelling and Responding to Effusive Eruptions*, 426 of *Geological Society, London, Special Publications*, Geological Society of London, 2016, pp. 387–408. <https://doi.org/10.1144/SP426.24>
- [42] J.M. Domínguez, A.J.C. Crespo, M. Gómez-Gesteira, Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method, *Comput. Phys. Commun.* 184 (3) (2013) 617–627. <https://doi.org/10.1016/j.cpc.2012.10.015>
- [43] U. Ghia, K.N. Ghia, C.T. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, *J. Comput. Phys.* 48 (3) (1982) 387–411. [https://doi.org/10.1016/0021-9991\(82\)90058-4](https://doi.org/10.1016/0021-9991(82)90058-4)
- [44] C.H. Marchi, C.D. Santiago, C.A.R. de Carvalho, Jr., Lid-driven square cavity flow: a benchmark solution with an 8192×8192 grid, *J. Verif. Valid. Uncert.* 6 (4) (2021) 041004. <https://doi.org/10.1115/1.4052149>
- [45] K. Venkatadri, S. Maheswari, C. Venkata Lakshmi, V. Ramachandra Prasad, Numerical simulation of lid-driven cavity flow of micropolar fluid, *IOP Conf. Ser.: Mater. Sci. Eng.* 402 (1) (2018) 012168. <https://doi.org/10.1088/1757-899X/402/1/012168>
- [46] G.Y. Dynnikova, Simulation of two-dimensional flow around an elliptical cylinder at high Reynolds numbers, *Phys. Fluids* 36 (2) (2024) 023109. <https://doi.org/10.1063/5.0189177>
- [47] W.F. Lindsey, Drag of cylinders of simple shapes, NACA Report 619, National Advisory Committee for Aeronautics, 1938.
- [48] N.K. Delany, N.E. Sorensen, Low-speed drag of cylinders of various shapes, NACA Technical Note 3038, National Advisory Committee for Aeronautics, 1953.
- [49] P. Sooraj, A. Sharma, A. Agrawal, Measurement of drag coefficient for an elliptical cylinder, *J. Energy Environ. Sustain.* 5 (2018) 1–7. <https://doi.org/10.47469/JEES.2018.v05.100050>
- [50] D. Durante, C. Pilloton, A. Colagrossi, Numerical simulations of the transition from laminar to turbulent regimes of planar viscous flows past airfoils, *Phys. Fluids* 36 (12) (2024) 124116. <https://doi.org/10.1063/5.0233990>
- [51] S. Marrone, A. Colagrossi, M. Antuono, G. Colicchio, G. Graziani, An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers, *J. Comput. Phys.* 245 (2013) 456–475. <https://doi.org/10.1016/j.jcp.2013.03.011>
- [52] E. Rossi, A. Colagrossi, D. Durante, G. Graziani, Simulating 2D viscous flow around geometries with vertices through the diffused vortex hydrodynamics method, *Comput. Methods Appl. Mech. Eng.* 302 (2016) 147–169. <https://doi.org/10.1016/j.cma.2016.01.006>
- [53] C. Suvanumrat, Comparison of turbulence models for flow past NACA0015 airfoil using OpenFOAM, *Eng. J.* 21 (3) (2017) 207–221. <https://doi.org/10.4186/ej.2017.21.3.207>
- [54] M.S. Selig, *Summary of Low Speed Airfoil Data*, 1, SoarTech Publications, Virginia Beach, VA, Virginia Beach, VA, 1995.
- [55] N. Prajapati, U. Patdiwala, P. Surani, Experimental validation of NACA 6409 airfoil using CFD analysis, *Int. J. Emerg. Technol. Innov. Res.* 8 (4) (2021) 757–764.

- [56] J.A. Fernández-Fernández, L. Westhofen, F. Löschner, S.R. Jeske, A. Longva, J. Bender, Fast octree neighborhood search for SPH simulations, *ACM Trans. Graph.* 41 (6) (2022) 242:1–242:13. <https://doi.org/10.1145/3550454.3555523>
- [57] R. Winchenbach, A. Kolb, Multi-level memory structures for simulating and rendering smoothed particle hydrodynamics, *Comput. Graph. Forum* 39 (6) (2020) 527–541. <https://doi.org/10.1111/cgf.14090>
- [58] L. Fu, Z. Ji, X.Y. Hu, N.A. Adams, Parallel fast-neighbor-searching and communication strategy for particle-based methods, *Eng. Comput.* 36 (3) (2019) 899–929. <https://doi.org/10.1108/EC-05-2018-0226>