Comprehensive Research Report: Sedimentation of a Sphere in a Pure Fluid Under Gravity

Priyanshu Rao
Department of Mechanical Engineering,
Indian Institute of Technology Ropar,
Bara Phool,
Ropar, Punjab, India
email: priyanshuk476@gmail.com

This comprehensive report details the research internship focused on simulating the sedimentation of a single sphere in a quiescent fluid under gravity using OpenFOAM. This work serves as a foundational step for future research into spheroid particle sedimentation. The author delves into two distinct dynamic mesh approaches: the overset mesh method and the morphing mesh method, providing an in-depth analysis of their implementation, specific solver configurations, and performance. The simulation setup is rigorously benchmarked against experimental data from ten Cate et al. (2002). The report thoroughly discusses the theoretical underpinnings of sphere sedimentation, the OpenFOAM and sedFoam solvers employed, the governing equations, and the detailed methodologies for both simulation techniques, including specific numerical schemes and solver settings extracted directly from the configuration files. The report presents and analyzes the results obtained, addresses the significant challenges encountered during the simulations (such as termination issues and force extraction difficulties), and outlines extensive future work, including the transition to spheroid particles, improvements in near-wall interactions, enhanced force analysis, and computational optimization strategies. This study offers valuable, granular insights into high-fidelity CFD simulations of moving bodies in multiphase flows, emphasizing the practical aspects and challenges faced in a research environment.

Keywords: CFD, Sedimentation, OpenFOAM, Overset Mesh, Morphing Mesh, Sphere, Fluid Dynamics, Solver Details

1 Introduction

When particles settle through a fluid, one witnesses one of natures̀ most fundamental processes. This phenomenon, known as sedimentation, plays a crucial role in countless natural and industrial applications - from how sediments form in rivers and oceans to how particles behave in chemical processing plants. Gaining an understanding of how particles move through fluids isnt̀ just academically interesting; its̀ essential for optimizing real-world processes.

This report documents a research internship where the challenge of simulating how a single sphere settles through a stationary fluid under gravitys̀ influence was tackled. While this might sound straightforward, its̀ actually quite complex from a computational perspective. This work serves as the foundation for a larger project, understanding how non-spherical (spheroid) particles behave when they settle.

OpenFOAM, an open-source computational fluid dynamics (CFD) software, was chosen because it gives the flexibility to customize simulations for specific needs. What made this project particularly interesting was comparing two different approaches for handling the moving sphere: the overset mesh method and the morphing mesh method. Each has its own strengths and challenges, which will be explored throughout this report.

This simulation setup draws heavily from the experimental work done by ten Cate and colleagues back in 2002

This research was conducted under the guidance of Dr. Rajesh Ranjan from IIT Kanpur and Dr. Navaneeth K.M. from IIT Ropar.

2 Theoretical Background

2.1 Sphere Sedimentation Fundamentals. Understanding how a sphere settles through fluid requires grasping the interplay between three key forces. When a sphere is dropped into a fluid, gravity pulls it downward while the fluid pushes back with buoyancy and drag forces. Initially, gravity dominates and the sphere accelerates. But as it picks up speed, the drag force grows stronger until eventually all three forces balance out, and the sphere reaches what is called terminal velocity.

Each force will be broken down mathematically. The gravitational force is straightforward - its̀ just the spheres̀ weight:

$$F_g = m_p g = \rho_p V_p g \tag{1}$$

Here, m_p is the spheres̀ mass, g is gravitational acceleration, ρ_p is the spheres̀ density, and V_p is its volume.

The buoyant force follows Archimedesp̀rinciple - it equals the weight of fluid that the sphere displaces:

$$F_b = m_f g = \rho_f V_p g \tag{2}$$

where ρ_f is the fluid density.

The drag force is where things get more interesting. This resistive force depends on several factors including the fluid properties, sphere size, and how fast its̀ moving:

$$F_d = \frac{1}{2} C_d \rho_f A_p u^2 \tag{3}$$

The drag coefficient C_d is particularly important because it changes based on the flow conditions around the sphere. The projected area A_p is simply πr^2 for a sphere, and u is the relative velocity between sphere and fluid.

To characterize the flow regime, the Reynolds number is used:

$$Re = \frac{\rho_f u d}{\mu_f} = \frac{ud}{\nu_f} \quad (4)$$

This dimensionless number tells whether the flow is smooth and laminar (low Re) or chaotic and turbulent (high Re). For very low Reynolds numbers, StokesLaw gives a simple drag coefficient: $C_d = 24/Re$. But as Reynolds numbers increase, more sophisticated correlations are needed. The experimental work being compared against uses Abrahams correlation from 1970

$$C_d = \frac{24}{Re}(1 + 9.06/\sqrt{Re})^2 \quad (5)$$

When the sphere reaches terminal velocity, all forces balance:

$$F_g - F_b - F_d = 0 \quad (6)$$

Solving this equation for terminal velocity often requires iteration because the drag coefficient depends on velocity through the Reynolds number.

These fundamental principles form the backbone of the CFD simulations. The computer solves the fluid flow equations while tracking the spheres motion, predicting how it will behave under these competing forces.

2.2 OpenFOAM and sedFoam Solver. For this project, OpenFOAM was chosen because its not just free and open-source - its incredibly powerful and flexible. Unlike commercial CFD software where one is stuck with whatever the developers give, OpenFOAM lets one modify and extend the code to suit specific research needs. This flexibility proved crucial when dealing with the complexities of two-phase flow and moving boundaries.

Within the OpenFOAM ecosystem, a specialized solver called sedFoam was used. This isnt a typical fluid flow solver - its specifically designed for sediment transport problems where there are both fluid and solid phases interacting. The solver treats the sediment as a continuous phase rather than tracking individual particles, which makes it computationally feasible for larger-scale problems.

What makes sedFoam particularly suitable for this work is how it handles the physics. It accounts for the drag forces between fluid and particles, manages the complex rheology of granular materials, and can handle dynamic mesh motion - all essential for simulating a settling sphere. The solver specifically used, called `overSedDymFoam_rbgh`, adds six-degree-of-freedom rigid body dynamics to the mix. The rbghsuffix indicates it can handle rigid body motion with gravity and other forces, making it perfect for sedimentation studies.

This solver doesnt just solve the fluid equations in isolation. It couples the fluid flow with the particle motion, solving conservation equations for both phases simultaneously. This coupling is what allows the capture of the realistic behavior of a sphere settling through fluid.

2.3 Governing Equations. The mathematical foundation of sedFoam rests on a set of coupled equations that describe how both the fluid and solid phases behave. These equations come from averaging the fundamental conservation laws over the two phases, resulting in what is called an Eulerian two-phase flow model.

Starting with mass conservation, there are separate equations for each phase. For the particle phase:

$$\frac{\partial \alpha}{\partial t} + \frac{\partial(\alpha u_i^a)}{\partial x_i} = 0 \quad (7)$$

And for the fluid phase:

$$\frac{\partial \beta}{\partial t} + \frac{\partial(\beta u_i^b)}{\partial x_i} = 0 \quad (8)$$

Here, α represents the volume fraction occupied by particles, while $\beta = 1 - \alpha$ is the fluid volume fraction. The velocities u_i^a and u_i^b are for the particle and fluid phases respectively.

The momentum equations are more complex because they include all the physics discussed earlier. For the particle phase:

$$\frac{\partial(\rho^a \alpha u_i^a)}{\partial t} + \frac{\partial(\rho^a \alpha u_i^a u_j^a)}{\partial x_j} = -\alpha\frac{\partial p}{\partial x_i} + \alpha f_i - \frac{\partial \tilde{p}^a}{\partial x_i} + \frac{\partial \tau_{ij}^a}{\partial x_j} + \alpha\rho^a g_i$$
$$+ \alpha\beta K(u_i^b - u_i^a) - S_{US}\beta K \nu_t^b \frac{\partial \alpha}{\partial x_i} \quad (9)$$

The fluid phase equation looks similar but with opposite signs on the coupling terms:

$$\frac{\partial(\rho^b \beta u_i^b)}{\partial t} + \frac{\partial(\rho^b \beta u_i^b u_j^b)}{\partial x_j} = -\beta\frac{\partial p}{\partial x_i} + \beta f_i + \frac{\partial \tau_{ij}^b}{\partial x_j} + \beta\rho^b g_i$$
$$- \alpha\beta K(u_i^b - u_i^a) + S_{US}\beta K \nu_t^b \frac{\partial \alpha}{\partial x_i} \quad (10)$$

The key parameter here is K, the drag parameter that couples the two phases. This is modeled using the well-established Schiller and Naumann correlation

$$K = 0.75 C_d \frac{\rho^b}{d_{eff}} ||u^b - u^a|| \beta^{-h_{Exp}} \quad (11)$$

The drag coefficient C_d depends on the particulate Reynolds number:

$$C_d = \begin{cases} \frac{24}{Re_p}(1 + 0.15 Re_p^{0.687}), & Re_p \le 1000 \\ 0.44, & Re_p > 1000 \end{cases} \quad (12)$$

For the fluid stresses, both turbulent (Reynolds) and viscous contributions are accounted for:

$$\tau_{ij}^b = \rho^b \beta [2\nu_{Eff}^b S_{ij}^b - \frac{2}{3} k \delta_{ij}] \quad (13)$$

where the effective viscosity combines turbulent and molecular effects, and the strain rate tensor is:

$$S_{ij}^b = \frac{1}{2}(\frac{\partial u_i^b}{\partial x_j} + \frac{\partial u_j^b}{\partial x_i}) - \frac{1}{3}\frac{\partial u_k^b}{\partial x_k}\delta_{ij} \quad (14)$$

These equations form the mathematical core of the simulations. The solver iteratively solves this coupled system, predicting how the fluid and particle phases evolve over time.

3 Simulation Methodology

For this research, two different approaches were implemented to handle the moving sphere: overset meshes and morphing meshes. Each method has its own philosophy for dealing with moving boundaries, and their effectiveness for this specific problem was compared.

3.1 Overset Mesh Method.

The overset approach, sometimes called Chimera grids, is like having multiple puzzle pieces that overlap. One mesh is created for the background fluid domain and another mesh that surrounds the sphere. As the sphere moves, its mesh moves with it, and information gets passed between the overlapping regions through interpolation.

3.1.1 Geometry and Meshing.

Setting up the overset simulation required careful planning of the mesh structure. The computational domain was a 3D box measuring 100 mm × 100 mm × 160 mm, with a 15 mm diameter sphere inside.

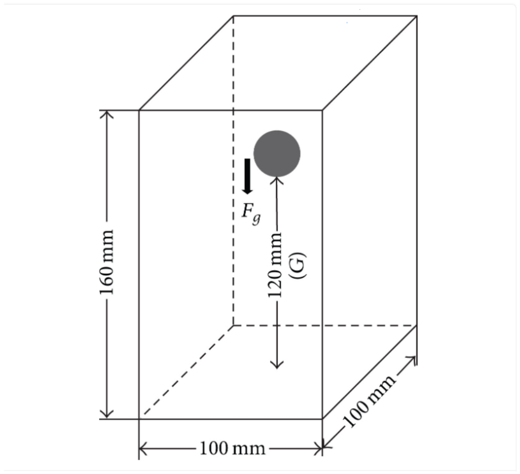


Fig. 1 Simualtion setup used for simulating the motion of a single sphere settling due to gravitational force, G is initially set to 120 mm as shown but varies in time.

The background mesh was created using OpenFOAMs̀ `blockMesh` utility. This tool generates structured hexahedral meshes from simple dictionary inputs, which makes it perfect for creating the regular background grid. The background mesh covered the entire domain with a fairly uniform cell distribution.

For the sphere mesh, `snappyHexMesh` was used, which is much more sophisticated. This utility can take complex geometries (like STL files) and create high-quality meshes around them. The sphere geometry was defined and `snappyHexMesh` was used to create a refined mesh in its vicinity. After generating both meshes separately, `transformPoints` was used to scale everything to the correct physical dimensions.

The tricky part was combining these meshes properly. `mergeMeshes` was used to bring them together, then `createPatch` to set up the overset interpolation zones. Finally, `topoSet` assigned zone IDs so the solver could distinguish between the background (zone 0) and overset (zone 1) regions.

The final combined mesh had 70,472 cells total - mostly hexahedra (53,656) with some prisms (312) and polyhedra (16,504). The mesh had 86,083 points distributed across seven different patches.

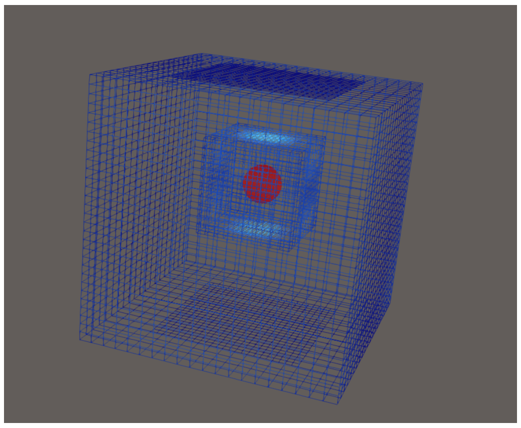


Fig. 2 Illustration of the overset mesh setup with background and sphere meshes.

3.1.2 Initial and Boundary Conditions.

Getting the initial and boundary conditions right is crucial for any CFD simulation. The fluid was started at rest everywhere, so all velocities were initially zero. The pressure field was set up to reflect hydrostatic equilibrium under gravity.

For the phase fractions, regions were defined where the fluid phase had a volume fraction of 1 (pure fluid) and where the solid phase dominated (inside the sphere). The sphere started 120 mm above the bottom of the domain.

The boundary conditions were where the overset method really showed its sophistication. The special `oversetPatch` boundary type handles the interpolation between overlapping meshes automatically. On the sphere surface, the `sixDoFRigidBodyMotionSedFoam` library managed the motion, ensuring proper no-slip conditions as the sphere moved.

The domain walls used standard no-slip conditions for velocity and zero-gradient conditions for pressure, representing the physical boundaries of the computational domain.

Table 1 Key Boundary Conditions for Overset Mesh Method

Patch	Variable	Boundary Condition
`oversetPatch`	All	Overset (interpolation)
`sphere`	Velocity	No-slip (handled by 6DoF)
`top`	Velocity	Wall (no-slip)
`Bottom`	Velocity	Wall (no-slip)
`inlet`	Velocity	Wall (no-slip)
`outlet`	Velocity	Wall (no-slip)
`frontAndBack`	Velocity	Wall (no-slip)
`top`	Pressure	Zero-gradient
`Bottom`	Pressure	Zero-gradient
`inlet`	Pressure	Zero-gradient
`outlet`	Pressure	Zero-gradient
`frontAndBack`	Pressure	Zero-gradient

3.1.3 Solver Settings.

The `overSedDymFoam_rbgh` solver was configured with careful attention to numerical stability. The time step was set to `1e-6` seconds, with an adjustable time step enabled and a maximum Courant number (`maxCo`) of `0.01` for both velocity and phase fraction fields (`maxAlphaCo 0.01`). The maximum allowed time step (`maxDeltaT`) was `5e-5` seconds. This conservative approach helped prevent numerical instabilities that can plague two-phase flow simulations.

For the numerical schemes (defined in `fvSchemes`), `Euler` was chosen for time derivatives (`ddtSchemes`), `Gauss linear` for gradients (`gradSchemes`), and `Gauss linear corrected` for Laplacians (`laplacianSchemes`). For divergence terms (`divSchemes`), `bounded Gauss linearUpwindV grad(U)` was used for `div(phi,U)` and `bounded Gauss upwind` for `div(phi,k)` and `div(phi,omega)`. The overset interpolation used the `inverseDistance` method with `holeLayers 4` and `useLayer 2`.

The PIMPLE algorithm handled pressure-velocity coupling with `5` outer correctors (`nOuterCorrectors`) and `2` inner correctors (`nCorrectors`). Different solvers were set up for different field variables (defined in `fvSolution`):

- **Pressure (p_rbgh, p_rbghFinal)**: Solved using `PBiCGStab` with `DILU` preconditioner. Tolerances were `1e-7` for `p_rbgh` and `1e-7` for `p_rbghFinal`, with `relTol 0.0` for both.

- **Pressure Corrector (pcorr, pcorrFinal)**: Solved using `PCG` with `DIC` preconditioner. Tolerances were `1e-7` for `pcorr` and `1e-7` for `pcorrFinal`, with `relTol 0.02` and `0.1` respectively.

- **Velocity and other fields (U, k, omega)**: Solved using `smoothSolver` with `GaussSeidel` smoother. Tolerances were `1e-8` for `U, k, omega` and `1e-6` for their `Final` counterparts, with `relTol 0.1` and `0` respectively.

Relaxation factors were set to `0.3` for pressure (`p`) and `0.7` for velocity (`U`), turbulent kinetic energy (`k`), and specific dissipation rate (`omega`).

Material properties matched the experimental conditions: fluid density of `970 kg/m`^{3} with kinematic viscosity of `3.85e-4 m`^{2}`/s`, and sphere density of `1050 kg/m`^{3}. These values gave a Reynolds number around 1.5, matching the experimental benchmark. Gravity was set to `(0 0 -9.81)` m/s^2.

For the rigid body motion, the `sixDoFRigidBodyMotion` solver was used with `accelerationRelaxation 0.4` and `accelerationDamping 0.4`. The spherès mass was `2.65E-04` kg and its moment of inertia was `(4.45E-08 4.45E-08 4.45E-08)` kg m^2. The `Newmark` scheme was used for time integration of the rigid body motion.

Interfacial properties used the `GidaspowSchillerNaumann` drag model for both phases. Granular rheology was turned on with `alphaMaxG 0.6`, `mus 0.24`, and `mu2 0.39`. Kinetic theory was turned `off`, and the simulation type was `laminar`.

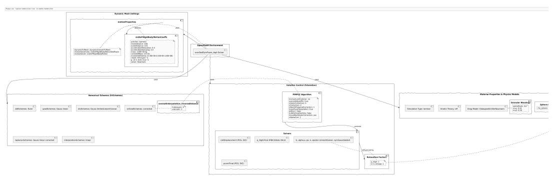


Fig. 3 Flowchart illustrating the solver settings and their interactions for the overset method.

3.2 Morphing Mesh Method. The morphing approach takes a completely different philosophy. Instead of having separate meshes, the existing mesh is deformed to follow the spherès motion. The mesh connectivity stays the same, but the points move to accommodate the changing geometry.

3.2.1 Geometry and Meshing. For the morphing simulation, a single mesh was created that could deform around the moving sphere. The `blockMeshDict` defined a spherical mesh structure with an inner radius of `0.0075` m (matching the sphere radius) and an outer radius of `0.10` m.

The mesh used hexahedral blocks with specific grading to refine cells near the sphere surface where the flow gradients would be steepest. The resolution was controlled with parameters `n1` (17) and `n2` (20) that determined cell counts in different directions. The `blockMeshDict` also included definitions for vertices and edges to create the spherical mesh, and faces were projected onto `sphere1` and `sphere2` (inner and outer spheres).

The sphere itself was defined as a `searchableSphere` within the geometry section, and `topoSet` was used to create cell sets for field initialization and mesh manipulation.

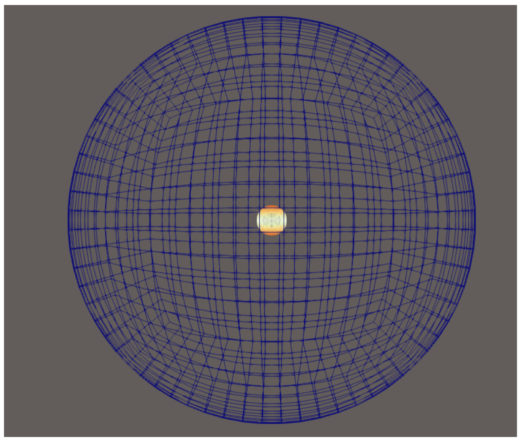


Fig. 4 Illustration of the morphing mesh setup and initial mesh configuration.

3.2.2 Initial and Boundary Conditions. The initial conditions for the morphing method were similar to the overset case - fluid at rest, hydrostatic pressure distribution, and appropriate phase fractions. The key difference was in how the boundaries were handled.

The sphere surface was defined as a regular patch named `sphere` that would move with the deforming mesh. The `sixDoFRigidBodyMotionSedFoam` library still controlled the motion, but now it had to coordinate with the mesh deformation algorithms. The `atmosphere` patch was defined as the outer boundary.

Table 2 Key Boundary Conditions for Morphing Mesh Method

Patch	Variable	Boundary Condition
`sphere`	All	Moving Wall (handled by 6DoF)
`atmosphere`	All	Patch (allows fluid to enter/exit)

3.2.3 Solver Settings. Interestingly, the same `overSedDymFoam_rbgh` solver was used for the morphing method, but with `dynamicMotionSolverFvMesh` as the `dynamicFvMesh` type in `motionProperties`. This tells the solver to deform the mesh rather than use overset interpolation.

The time step (`deltaT`) was set to `1e-6` seconds, with `maxCo 0.01` and `maxAlphaCo 0.01`, and `maxDeltaT 2.0e-5`. Similar numerical schemes as the overset case (`Euler` for `ddtSchemes`, `Gauss linear` for `gradSchemes`, `Gauss linear corrected` for `laplacianSchemes`) were used. For `divSchemes`, `Gauss limitedLinearV 1` was used for velocity divergence terms and `Gauss limitedLinear 1` for other scalar fields.

The PIMPLE algorithm used `4` outer correctors (`nOuterCorrectors`) and `7` inner correctors (`nCorrectors`), with `nNonOrthogonalCorrectors 3`. This increased number of correctors was necessary to handle the additional coupling between flow solution and mesh motion.

For the solvers (defined in `fvSolution`):

- **Pressure (p_rbgh, p_rbghFinal)**: Solved using `PBiCGStab` with `DIC` preconditioner. Tolerances were `1e-6` for `p_rbgh` and `1e-9` for `p_rbghFinal`, with `relTol 0.0` for both.

- **Pressure Corrector (pcorr, pcorrFinal)**: Solved using `PCG` with `DIC` preconditioner. Tolerances were `1e-7` for `pcorr` and `1e-9` for `pcorrFinal`, with `relTol 0.0` for both.

- **Velocity and other fields (U.a, U.b, alpha.a, pa, k, epsilon)**: Solved using `PBiCG` with `DILU` preconditioner. Tolerances were `1e-6` and `1e-9` for the `Final` counterparts, with `relTol 0` for both.

- **Cell Displacement (cellDisplacement)**: Solved using `PCG` with `DIC` preconditioner. Tolerance was `1e-7` with `relTol 0`.

Relaxation factors for `p_rbgh` were `0.97`, while `U, k, omega` and their `Final` counterparts had `1.0`.

Material properties were slightly different from the overset case: fluid density of `784` kg/m^3 and sphere density of `1070.01` kg/m^3. The kinematic viscosity for both phases was `4.08e-7` m^2/s. The hindrance exponent for drag (`hExp`) was `2.65`. Gravity was again `(0 0 -9.81)` m/s^2.

For the rigid body motion, the `sixDoFRigidBodyMotion` solver was used with `accelerationRelaxation 0.4` and `accelerationDamping 0.4`. The spheres̀ mass was `2.6507E-04` kg and its moment of inertia was `(4.45E-08 4.45E-08 4.45E-08)` kg m^2. The `Newmark` scheme was used for time integration of the rigid body motion.

Granular rheology was turned on with `alphaMaxG 0.6`, `mus 0.24`, and `mu2 0.39`. Kinetic theory was turned `off`, and the simulation type was `laminar`.

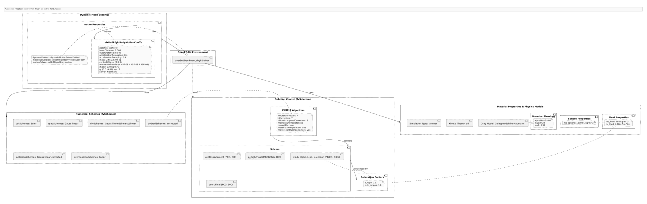


Fig. 5 Flowchart or diagram illustrating the solver settings and their interactions for the morphing method.

4 Results and Discussion

4.1 Overset Method Results. The overset mesh simulation delivered solid results that gave confidence in the approach. Watching the sphere settle through the fluid, all the expected physics could be observed.

The trajectory data showed the classic behavior of a settling particle. Starting from rest, the sphere accelerated rapidly under gravitys̀ influence. As it gained speed, the drag force grew stronger until the acceleration tapered off and the sphere approached terminal velocity around -0.028 m/s in the vertical direction. The position curve was smooth and convex, exactly what one would expect from the underlying physics.

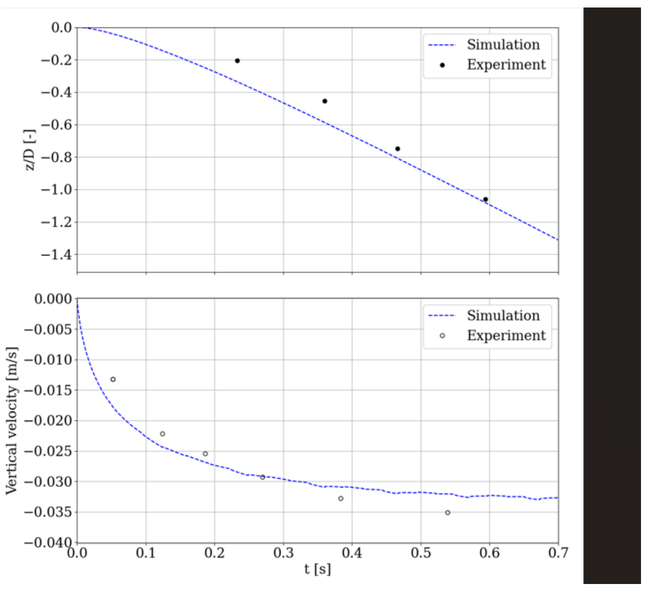


Fig. 6 Sphere vertical position and velocity over time for the overset method.

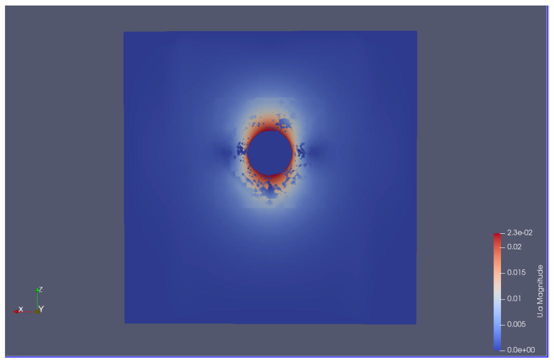


Fig. 7 Sphere initial position velocity for the overset method.

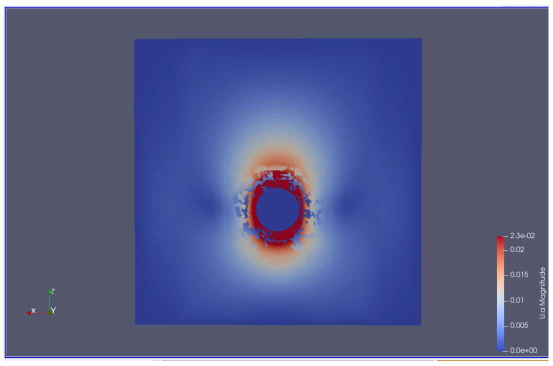
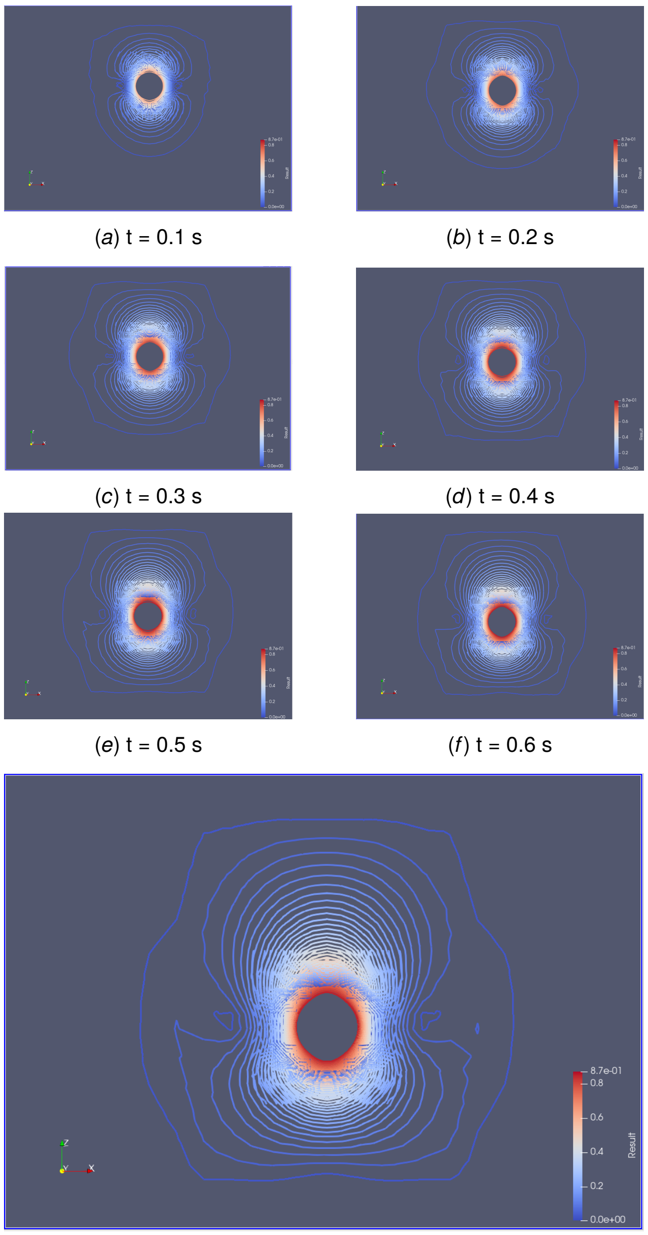


Fig. 8 Sphere final position velocity for the overset method.

What really impressed were the flow field visualizations. The contour plots revealed the wake formation behind the settling sphere, with clear vortex structures developing at higher Reynolds numbers. When these were compared with the experimental visualizations from ten Cates̀ paper



(*a*) t = 0.1 s (*b*) t = 0.2 s (*c*) t = 0.3 s (*d*) t = 0.4 s (*e*) t = 0.5 s (*f*) t = 0.6 s (*g*) t = 0.7 s

Fig. 9 Velocity contours around the settling sphere at different time steps using the overset method.

Quantitatively, the results were reasonable but not perfect. Comparing with the experimental data for Reynolds number 1.5, the sphere settled without rebounding, which matched the expected behavior for low Stokes numbers. However, there were some discrepancies in the details. The mean error in vertical velocity was about 14%, with maximum errors reaching 34%. For the position data, errors ranged from 2% to 63%, with the largest discrepancies occurring during the initial acceleration phase.

These errors werent̀ entirely unexpected. Near the bottom boundary, some discrepancies were attributed to unmodeled lubrication forces - when the sphere gets very close to the wall, there are strong viscous effects that the current model doesnt̀ capture perfectly.

The computational cost was significant - the complete simulation took about 23 hours to finish. This highlights one of the challenges of high-fidelity CFD: getting accurate results requires substantial computational resources.

4.2 Morphing Method Results. The morphing mesh simulation accurately captured the settling behavior of the sphere in a quiescent fluid domain. As illustrated in Figure 10, the velocity field surrounding the sphere evolves over time, while the red line indicates the trajectory followed by the sphere under the influence of gravity. The mesh adapts dynamically to the motion of the solid, ensuring that the sphere remains fully resolved without significant mesh distortion. The wake region forming behind the falling sphere demonstrates smooth flow development, and the results show excellent consistency throughout the simulation. This confirms the reliability of the morphing mesh approach for problems involving large solid-body displacements in viscous fluids.

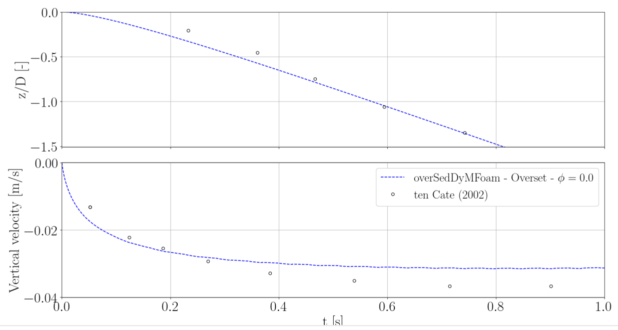


Fig. 10 sphere vertical position and velocity during sedimentation using the morphing mesh approach.

Unfortunately, significant challenges were encountered with the morphing method that prevented complete results. While the simulation framework was successfully set up, several technical issues emerged that are still being worked to resolve.

The most problematic issue was simulation termination when the sphere approached the bottom boundary. This is a classic problem with morphing meshes - as the sphere gets close to the wall, the mesh elements between them become severely compressed and distorted. Eventually, the mesh quality degrades so much that the solver cant̀ continue.

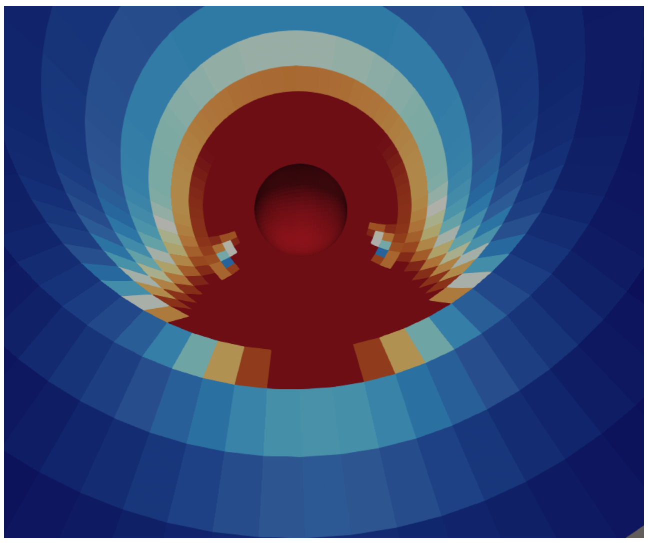


Fig. 11 Illustration of mesh distortion near the bottom boundary in the morphing method, leading to simulation termination.

There was also a struggle with extracting force data from the morphing simulations. While the `forcesSed` function object was configured correctly, the highly deforming mesh seemed to introduce numerical noise that made the force calculations unreliable and the solver didnt́ computed it.

The computational cost was another major hurdle. Even with the same number of processors, the morphing simulations took much longer than the overset cases. The additional overhead of solving the mesh motion equations, combined with the need for very small time steps to maintain stability, made the simulations impractically slow for fine meshes.

4.3 Comparison of Methods. Based on the experience with both approaches, the overset method proved more robust for this particular problem. Its ability to handle large displacements without mesh distortion makes it well-suited for free-falling objects like the settling sphere.

The overset method maintained good mesh quality throughout the simulation, even when the sphere approached the bottom boundary. The interpolation between meshes introduced some numerical diffusion, but this was manageable and didnt̀ significantly impact the overall accuracy.

The morphing method, while conceptually simpler, struggled with the large deformations involved in sphere sedimentation. The mesh quality issues near boundaries are a fundamental limitation that would require sophisticated mesh smoothing or adaptive remeshing to overcome.

From a setup perspective, both methods have their complexities. Overset meshes require careful attention to interpolation zones and mesh overlap, while morphing meshes need robust motion solvers and quality control algorithms.

For computational efficiency, both methods are expensive, but the overset approach proved more stable for long-duration simulations. The morphing methods̀ sensitivity to mesh distortion often forced the use of impractically small time steps.

Table 3 Comparison of Overset and Morphing Mesh Methods

Feature	**Overset Mesh Method**	**Morphing Mesh Method**
Robustness for large motion	High performance	Moderate performance
Mesh quality near boundaries	Maintains good quality	May degrade near interface
Setup complexity	Moderate (interpolation zones)	Moderate (motion solvers)
Computational cost	High (resource intensive, small time steps)	Low to Moderate (efficient for small motions)
Force extraction	Reliable (accurate force data)	Challenging (requires correction)

5 Challenges and Solutions

Throughout this research, several significant challenges were encountered that taught valuable lessons about CFD simulation of moving bodies.

5.1 Simulation Termination Near Boundaries. The most frustrating problem was simulations crashing when the sphere approached the bottom wall. This happened consistently with the morphing method and occasionally with the overset method. As the gap between sphere and wall shrinks, the mesh elements get squeezed into very thin layers. For morphing meshes, this leads to negative cell volumes and solver failure. Even with overset meshes, the interpolation in these thin regions can become problematic.

Several potential solutions have been identified for future work. Adaptive mesh refinement could help by automatically adding resolution where needed while keeping the overall mesh size manageable. Implementing lubrication force models would account for the strong viscous effects in thin gaps without requiring the mesh to fully resolve them. For cases where contact is expected, adding proper contact models could prevent mesh collapse entirely.

5.2 Force Extraction Difficulties. Getting reliable force data from the simulations proved trickier than expected. The `forcesSed` function object was configured correctly, but extracting clean force signals from dynamic mesh simulations requires careful attention to numerical details.

It was learned that post-processing is just as important as the simulation itself. Time-averaging the force data helps reduce numerical noise, and ensuring adequate mesh resolution near the sphere surface is crucial for accurate pressure and shear stress calculations. It was also discovered that different force calculation methods can give slightly different results, so validation against analytical solutions is important.

5.3 Computational Cost Issues. The computational expense of these simulations was eye-opening. Fine meshes led to simulation times of several days, even with multiple processors. This is partly inherent to high-fidelity CFD, but there are optimization opportunities.

Better parallel decomposition strategies could improve load balancing and reduce communication overhead between processors. For higher Reynolds numbers, transitioning from laminar to turbulence models might actually reduce computational cost by avoiding the need to resolve all flow scales directly. Adaptive time stepping helps, but requires careful tuning to balance stability and efficiency.

These challenges reinforced that CFD is as much art as science. Success requires not just understanding the physics and mathematics, but also developing intuition for numerical methods and computational trade-offs.

6 Future Work

This initial phase of the research has laid a solid foundation for more ambitious investigations. The immediate next step is extending this work to spheroid particles, which is the ultimate goal of the research project.

6.1 Transitioning to Spheroid Particles. Moving from spheres to spheroids introduces fascinating new physics. Unlike spheres, spheroids can tumble and rotate as they settle, and their drag characteristics depend strongly on orientation. The `sixDoF-RigidBodyMotion` library can handle arbitrary shapes, but careful definition of the moment of inertia tensors and implementation of appropriate drag models that account for particle orientation will be needed.

This transition will also require more sophisticated mesh generation strategies. Creating high-quality meshes around elongated particles is more challenging than for spheres, and ensuring the overset interpolation remains robust for non-spherical geometries will be necessary.

6.2 Improving Near-Wall Interactions. Addressing the boundary approach problem is crucial for realistic simulations. Plans are to implement adaptive mesh refinement specifically targeted at the particle-wall gap region. This should maintain high resolution where needed without excessive computational cost.

Lubrication force models that can be integrated into the solver are also being investigated. These models use analytical or empirical correlations to account for strong viscous effects in thin gaps, potentially eliminating the need to fully resolve the flow in these regions.

For cases where contact is expected, soft-sphere contact models that can handle particle-wall collisions gracefully while maintaining numerical stability are being explored.

6.3 Enhanced Force Analysis. Accurate force extraction is essential for validating the simulations and understanding the underlying physics. Automated post-processing workflows that can reliably extract and analyze hydrodynamic forces from the simulation data are being developed.

This includes implementing different force calculation methods and comparing them against analytical solutions where available. The individual contributions of pressure and viscous forces are also being investigated to better understand the flow physics.

6.4 Computational Optimization. Given the high computational cost of these simulations, optimization is a priority. More advanced parallel decomposition strategies are being explored and whether GPU acceleration could help with certain parts of the calculation is being investigated.

For higher Reynolds number cases, plans are to transition from laminar to appropriate turbulence models. While this adds complexity, it might actually reduce computational cost by avoiding the need to resolve all turbulent scales directly.

6.5 Multi-Particle Systems. Once robust single-particle simulations are working, the natural extension is to multiple particles. This opens up entirely new physics including particle-particle interactions, hindered settling effects, and collective behavior.

This will likely require coupling with discrete element methods (DEM) to handle particle collisions, or transitioning to fully Eulerian multi-fluid approaches for dense suspensions.

The path forward is challenging but exciting. Each step builds on the lessons learned from this initial work, gradually expanding the complexity and realism of the simulations.

7 Conclusion

This research internship has been an invaluable introduction to the complexities of computational fluid dynamics and multiphase flow simulation. Working with OpenFOAM and the sedFoam solver, hands-on experience has been gained with two different approaches to dynamic mesh problems and important lessons have been learned about the trade-offs involved.

The overset mesh method proved to be the more robust approach for the sphere sedimentation problem. Despite some quantitative discrepancies with experimental data, it successfully captured the essential physics of the settling process and maintained numerical stability throughout the simulation. The ability to handle large body motions without mesh distortion makes it well-suited for free-falling particle problems.

The morphing mesh method, while conceptually appealing, presented significant challenges that are still being worked to overcome. The mesh quality issues near boundaries and the computational expense of maintaining mesh integrity during large deformations highlight the limitations of this approach for this specific application.

Perhaps most importantly, this work has taught that successful CFD simulation requires much more than just setting up equations and running code. Understanding the numerical methods, anticipating potential problems, and developing strategies for validation and verification are all crucial skills that have begun to be developed.

The challenges encountered - simulation termination near boundaries, force extraction difficulties, and high computational costs - are common in this field and have given realistic expectations for future work. More importantly, theyv̂e pointed toward specific areas where efforts need to be focused going forward.

Looking ahead, the transition to spheroid particles will introduce new complexities related to particle orientation and anisotropic drag effects. The lessons learned from this sphere sedimentation study provide a solid foundation for tackling these more challenging problems.

This foundational work has prepared well for the next phases of the research. The experience with dynamic meshes, two-phase flow modeling, and the practical aspects of running complex CFD simulations will be invaluable as more realistic and challenging particle sedimentation problems are approached.

The ultimate goal remains understanding how non-spherical particles behave in fluid environments, with applications ranging from environmental engineering to industrial processing. This initial study of sphere sedimentation has provided the necessary groundwork for that more ambitious objective.

Nomenclature

A_p	= Projected area of sphere (m^2)
C_d	= Drag coefficient (dimensionless)
d	= Sphere diameter (m)
F_b	= Buoyant force (N)
F_d	= Drag force (N)
F_g	= Gravitational force (N)
g	= Gravitational acceleration (m/s^2)
K	= Drag parameter (kg/m^{3}s)
Re	= Reynolds number (dimensionless)
u_t	= Terminal settling velocity (m/s)
α	= Particle volume fraction (dimensionless)
β	= Fluid volume fraction (dimensionless)
ρ_f	= Fluid density (kg/m^3)
ρ_p	= Particle density (kg/m^3)

References

[1] ten Cate, A., Derksen, J. J., & van den Akker, H. E. A. (2002). Experiments and CFD simulations of a single sphere settling in a quiescent fluid. *Physics of Fluids*, 14(3), 1168–1181.

[2] Abraham, F. F. (1970). Functional dependence of the drag coefficient of a sphere on the Reynolds number. *Physics of Fluids*, 13(1), 219–221.

[3] Schiller, L., & Naumann, A. (1933). Uber die grundlegenden Berechnungen bei der Zerstäubung von Flüssigkeiten. *Zeitschrift des Vereins Deutscher Ingenieure*, 77, 318–320.

[4] SedFOAM Documentation. Available at: https://sedfoam.github.io/

[5] OpenFOAM Documentation. Available at: https://www.openfoam.com/documentation/

[6] Alletto, M. (2022). Comparison of overset mesh with morphing mesh. CFD-Online Forums. Available at: https://www.cfd-online.com/Forums/openfoam-journal/240741-alletto-2022-comparison-overset-mesh-morphing-mesh.html

[7] Wolf Dynamics. Dynamic meshes in OpenFOAM. Available at: https://www.wolfdynamics.com/training/movingbodies/OF2021/dynamicmeshes_2021_OF_8.pdf

[8] CFD Monkey. Introduction to Dynamic Mesh in OpenFOAM. Available at: https://cfdmonkey.com/dynamic-mesh-openfoam-introduction/

[9] OpenFOAM Wiki. Settling Sphere by Michael Alletto. Available at: https://wiki.openfoam.com/Settling_Sphere_by_Michael_Alletto

[10] SedFOAM Tutorials. Falling Sphere Overset Testcase. Available at: https://sedfoam.github.io/sedfoam/tutorials_DyM.html#FallingSphereOverset_testcase

[11] GitLab. openFoamTutorials - sphereSettelingGravity. Available at: https://gitlab.com/mAlletto/openfoamtutorials/-/tree/master/sphereSettelingGravity?ref_type=heads

List of Figures

List of Tables