Harnessing compute shaders for the real-time simulation of physically accurate, high-energy fluid dynamics

Evelyn Moore 03/2025 s4104592

Contents

1.0 Introduction………………………………………………………………………………….2

2.0 Literature Review…………………………………………………………………………...4

2.1 Fluid Simulation Techniques……………………………………………………….4

2.2 Numerical Integration Methods……………………………………………………6

2.2.1 Integration for Navier-Stokes approximation…………………………..6

2.2.2 Explicit integration………………………………………………………..6

2.2.3 Implicit integration………………………………………………………...7

2.2.4 Hybrid integration methods……………………………………………...7

2.3 Compute Shaders…………………………………………………………………..8

2.3.1 Compute shaders for explicit integration……………………………….8

2.3.2 Compute shaders for implicit integration……………………………….9

2.3.3 Compute shaders for hybrid integration methods…………………….9

2.4 Common Artifacts in High-Energy Fluid Simulation…………………………...11

2.4.1 Energy non-conservation……………………………………………….11

2.4.2 Excessive damping……………………………………………………...11

2.4.3 Exaggerated splashes………………………………………………….11

2.4.4 Temporal Aliasing……………………………………………………….12

2.4.5 Grid-aligned artifacts……………………………………………………12

2.5 Evaluating Stability and Realism………………………………………………...12

2.5.1 Stability…………………………………………………………………...12

2.5.2 Realism…………………………………………………………………..13

2.5.3 Qualitative benchmarks………………………………………………...13

2.5.4 Quantitative benchmarks……………………………………………….13

2.5.5 Physical benchmarks………………………………………………...…13

2.5.6 Performance benchmarks……………………………………………...14

2.6 Conclusion…………………………….…………………………………………...14

3.0 Output Design……………………………………………………………………………..16

3.1 Summary…………………………………………………………………………...16

3.2 Eulerian simulation………………………………………………………………..16

3.3 Lagrangian simulation…………………………………………………………….17

3.4 Metrics……………………………………………………………………………...17

3.5 Expected outcomes……………………………………………………………….17

4.0 References…………………………………………………………………………………18

1.0 Introduction

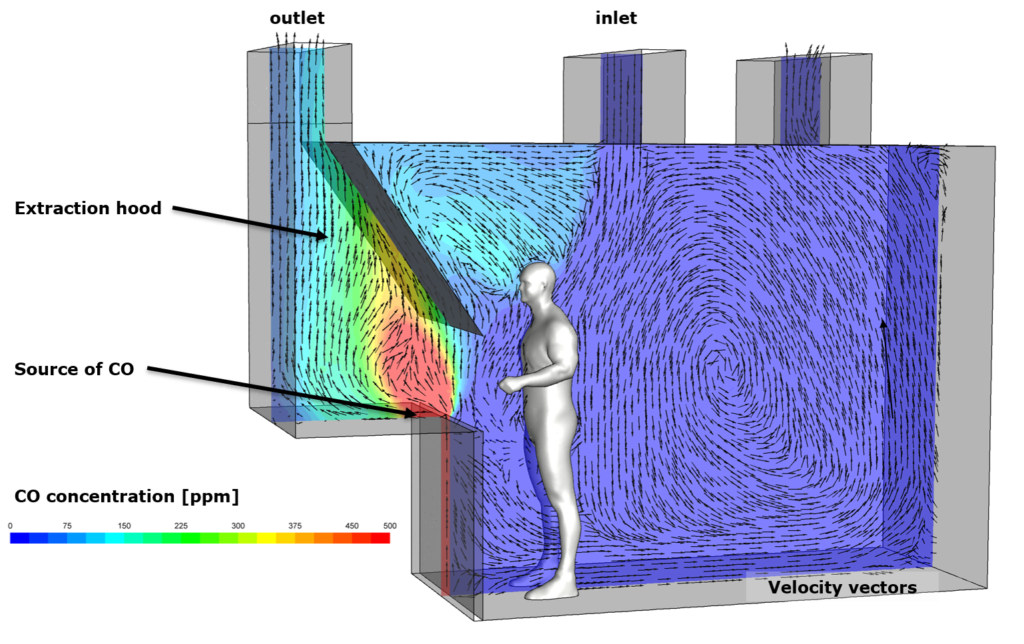
Fluids are complex, chaotic systems. They consist of a huge number of particles that interact through intermolecular forces, and very small changes to the initial conditions of a fluid can result in vastly different outcomes (Pazmino. S. A., 2024). Fluid simulation is vital in many fields, including environmental modelling, climate science, aerodynamics, medicine, film, and video games (Borthakur & Sarmah, 2013). However, in many of these cases accuracy is of the utmost importance, and therefore real-time simulations are inadequate. The main use case for real-time fluid simulation is in video games, because they are interactive environments where players can interact with the simulation in unexpected ways.

Figure 1.0 A fluid simulation to virtually test the safety of extraction hood ventilation designs (Image: InsPyro 2019)

Maintaining both physically accurate behaviour and real-time performance at the same time requires a delicate balance between computational efficiency, numerical stability, and visual realism. As the appetite for immersive, visually stunning experiences grows over time (Markopoulos, 2019), so too does the need for innovative approaches to fluid simulation in games that can still operate within the stringent performance requirements for achieving real-time framerates.

This paper explores the use of compute shaders as the primary tool for implementation of highly parallel solutions for high-energy fluid dynamic simulations. Compute shaders, which harness the immense computing potential of modern GPU chips, allow the calculation of approximations to the Navier-Stokes equations at real-time speeds (Aissa, 2017) (Schreiber & Neumann, 2010). These equations, which govern fluid motion, often lack analytical solutions, so approximations must be made instead (Iskhakov & Dinh, 2020). Many numerical integration methods exist for fluid simulation calculations, and this paper will investigate the feasibility of several of these methods for different situations. Specifically, the trade-offs between explicit and implicit integration methods will be evaluated, with a focus on their suitability for implementation within compute shaders. Additionally, Eulerian and Lagrangian spatial discretization approaches will be compared in the context of real-time simulation for video games.

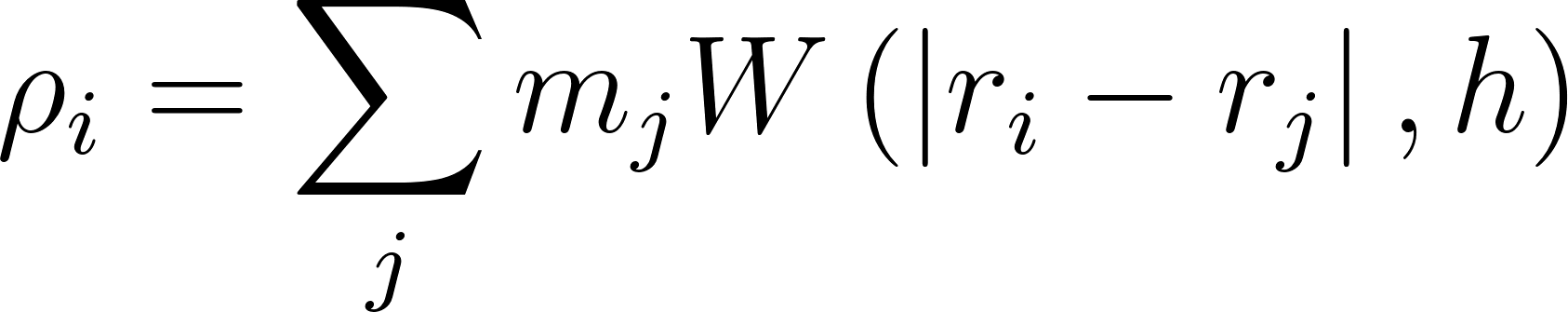
Through a thorough review of existing literature and practical examples from the gaming industry, this paper aims to highlight the strengths and weaknesses of different approaches, with respect to real-time interactive environments. Ultimately, this research will provide a roadmap for developers and researchers by focusing on practical applications of different numerical integration methods and spatial discretization options for fluid simulation.

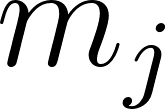
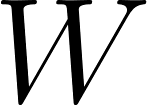
2.0 Literature Review

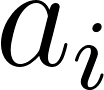
2.1 Fluid Simulation Techniques

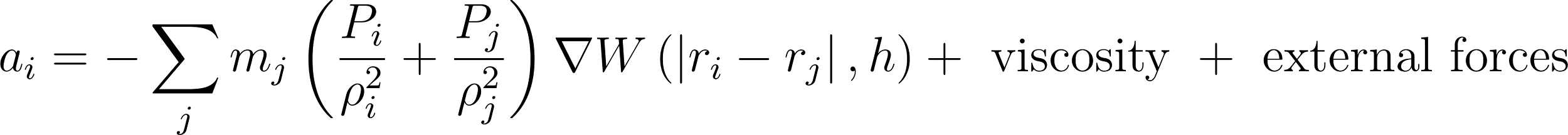
In 1822 Claude Navier introduced the concept of viscosity into the earlier Euler equations for fluid motion, and in 1845 George Stokes further refined the equations, giving rise to the famous Navier-Stokes equations that accurately describe the motion of fluid substances (Batchelor, 2000). The equations describe the motion of fluids by expressing the conservation of mass and momentum within a flow field (Hirom, 2024). A flow field is a way to represent fluid motion, involving the assignment of velocity vectors to each point in a finite region. In between these points, the velocity of the fluid can be calculated by interpolating between the values at the nearest points. The Navier-Stokes equations, which take the form of partial differential equations, model the relationship between velocity, pressure, density, and external forces acting on a fluid (Horváth, 2012).

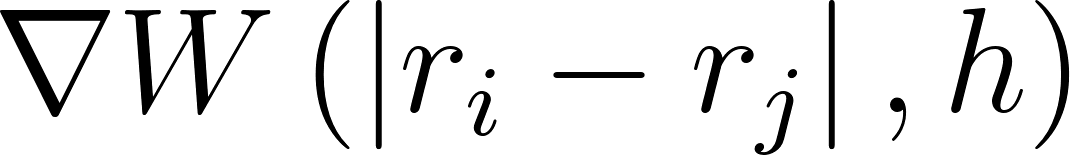
Solving these equations computationally requires a technique to approximate fluid behavior, typically this will be either a particle-based (Lagrangian) or grid-based (Eulerian) approach (Wang, Sun, & Hu, 2025). A frequently used Lagrangian approach called Smoothed Particle Hydrodynamics (SPH) treats fluids like a collection of discrete particles. Each particle has properties such as density, pressure, and velocity, and these properties influence the surrounding particles according to a smoothing function, typically a Cubic Spline kernel, also known as a Gaussian kernel (Wang, Sun, & Hu, 2024). The density [](https://www.codecogs.com/eqnedit.php?latex=%5Crho_i#0) at each particle is estimated by summing the contributions of its neighboring particles:



Where [](https://www.codecogs.com/eqnedit.php?latex=m_j#0)​ is the mass of the neighbouring particle [](https://www.codecogs.com/eqnedit.php?latex=j#0), [](https://www.codecogs.com/eqnedit.php?latex=r_i-r_j#0)​ is the distance between particles, and [](https://www.codecogs.com/eqnedit.php?latex=W#0) is the smoothing kernel function.

The acceleration [](https://www.codecogs.com/eqnedit.php?latex=a_i#0) of a particle is computed as:



Where [](https://www.codecogs.com/eqnedit.php?latex=%5Crho_i#0) and [](https://www.codecogs.com/eqnedit.php?latex=%5Crho_j#0)​ are the pressures of particles [](https://www.codecogs.com/eqnedit.php?latex=i#0) and [](https://www.codecogs.com/eqnedit.php?latex=j#0), and The term [](https://www.codecogs.com/eqnedit.php?latex=%5Cnabla%20W%5Cleft(%5Cleft%7Cr_i-r_j%5Cright%7C%2C%20h%5Cright)#0) represents the gradient of the smoothing kernel function. The equation ensures momentum conservation, which is critical for maintaining physically accurate fluid motion (Wang, Sun, & Hu, 2024).

SPH approaches allow for complex geometries and adaptive resolution, which makes it applicable for high detail applications, however the computational cost is high due to the number of neighbour searches required, and it is difficult to enforce incompressibility (Huang, 2024). In contrast, Eulerian approaches such as those using Marker-And-Cell (MAC) grids naturally display incompressibility, because they are calculated on a fixed spatial lattice. They efficiently solve the Navier-Stokes equations within a finite volume (Horváth, 2012). Eulerian methods excel at large-scale simulations such as weather and ocean current modelling, but can be prone to unrealistic damping and energy loss. They can be thought of as providing a picture of the spatial distribution of fluid velocity (and of other flow quantities such as density and pressure) at each instant during the motion. Instead of keeping track of numerous moving particles, many static points laid out in a grid throughout the simulation volume are used to store the flow quantities at those positions. (Batchelor, 2000). Research in recent years has sought to combine the computation efficiency of the Eulerian model with the accuracy and flexibility of the Lagrangian model, into a hybrid method. (Koumoutsakos & Cottet, 2009). One of the most widely used fluid simulation methods in computer graphics and video games is the famous FLIP (FLuid Implicit Particles) method (Brackbill and Ruppel. 1986) which effectively implements a hybrid model for significant advantages over both Eulerian and Lagrangian methods, and is used frequently in fluid animation. (Mastin, Watterberg & Mareda, 1987).

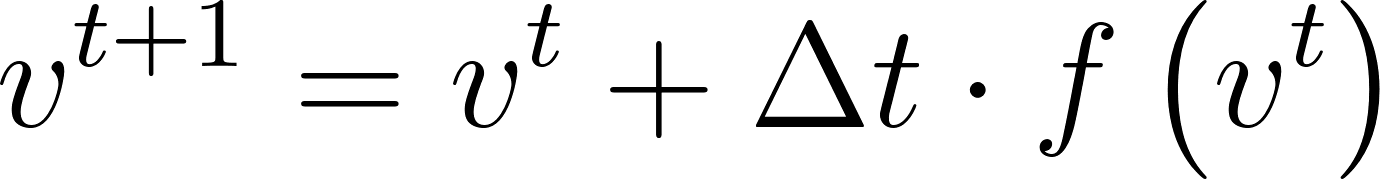
2.2 Numerical Integration Methods

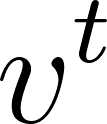
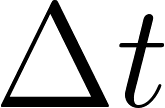
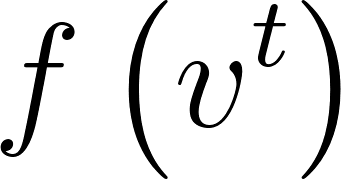
2.2.1 Integration for Navier-Stokes Approximation

Numerical integration methods are techniques to approximate solutions to the Navier-Stokes equations. Fluid properties such as velocity, pressure, and density are calculated over discrete time steps (in the case of video games, “frames” or “ticks”). Since the Navier-Stokes equations do not always have analytical solutions, developers must make a decision as to the type of numerical integration method is most suited to their use case. There are three main types of method that have relevance to fluid simulation - Explicit integration, Implicit integration, and hybrid methods which combine the two (Iskhakov & Dinh, 2020).

2.2.2 Explicit integration

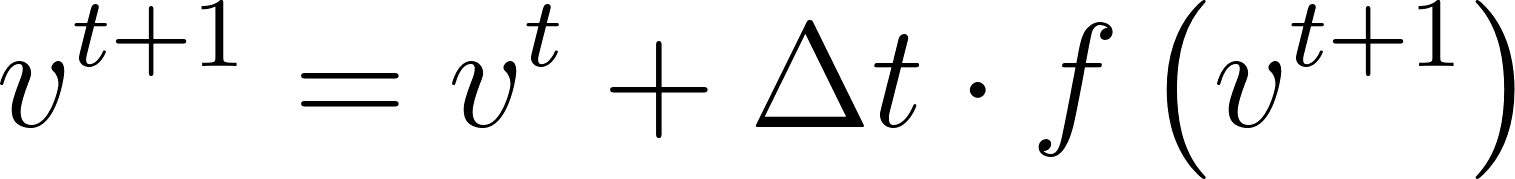
Explicit methods compute the next state of the simulation using only the current state. They are prone to numerical instability, which can cause fluid explosions, and over time can cause energy loss or growth because of the numerical errors accumulating. They are susceptible to the Courant-Friedrichs-Lewy condition, meaning they can require very small timesteps to maintain accuracy in some situations (Eisemann et al., 2013). Their computational efficiency makes explicit methods ideal for real-time simulation in video games. Euler's Explicit Method is used:

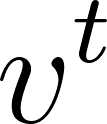
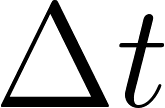


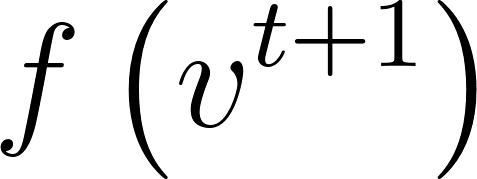
where [](https://www.codecogs.com/eqnedit.php?latex=v%5Et#0) is velocity at time [](https://www.codecogs.com/eqnedit.php?latex=t#0), [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%20t#0) is the time step, and [](https://www.codecogs.com/eqnedit.php?latex=f%5Cleft(v%5Et%5Cright)#0) represents forces (e.g., pressure, gravity) (Adegboye, Danjuma & Aniki, 2023).

2.2.3 Implicit integration

Implicit methods compute the next state of the simulation based on current and future information. Implicit solvers are very accurate, but with a high computational cost, which makes them ideal for scientific applications but generally unsuitable for real-time applications such as video games, unless precalculated and baked into a premade effect (Wang, 2024). When implementing an implicit solver, the Backward Euler Method is used instead of Euler’s Explicit Method:



Where [](https://www.codecogs.com/eqnedit.php?latex=v%5Et#0) is velocity at time [](https://www.codecogs.com/eqnedit.php?latex=t#0), and [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%20t#0) is the time step (Adegboye, Danjuma & Aniki, 2023).

[](https://www.codecogs.com/eqnedit.php?latex=f%5Cleft(v%5E%7Bt%2B1%7D%5Cright)#0) depends on the future state of the simulation, which requires a matrix solver to calculate due to the large systems of linear equations involved (Va et al, 2022). There are many types of matrix solvers, such as direct and iterative methods. Direct methods are very accurate but computationally expensive whereas iterative methods are only approximations but as suitable for large scale simulations. Implicit integration methods are numerically stable but computationally expensive, and can introduce damping which will mask small details, for example in turbulent areas (Shaffer, 1992).

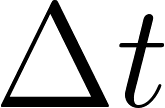
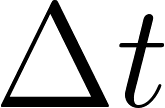
2.2.4 Hybrid integration methods

Hybrid methods include any approach which attempts to combine the efficiency of explicit methods and the stability of implicit methods. Usually, these “semi-implicit” implementations involve two steps - first, an explicit process is used to estimate a velocity, and then an implicit step is used to validate or correct the velocity (Khedkar, 2018, p.5-7). This “Predictor-Corrector” cycle ensures that the simulation remains stable even over long periods of time where numerical errors would otherwise slowly stack up. Hybrid methods typically preserve fine details better than implicit methods, although with less stability in extreme scenarios. They achieve computational speeds close to those of explicit methods (Nabizadeh et al. 2024). They are suited to applications such as engineering applications, where both accuracy and speed are required.

2.3 Compute Shaders

Fluid simulations are inherently suited to parallel computing architectures due to the huge number of similar calculations taking place. Simulations typically involve computing interactions across millions of particles or grid points, with numerous variables updated at each time step. Compute shaders, which are shader programs designed for arbitrary computation rather than handling traditional shader information such as vertices or fragments, execute programs directly on the GPU (Giner et al, 2024). Modern fluid simulations overwhelmingly leverage the power of Graphics Processing Units (GPUs), because their massively parallel architecture is more suitable for this type of large workload over CPU based solvers (Kass & Miller, 1990). This enables significant performance gains over traditional CPU-based implementations, making them the standard for high-performance fluid simulations.

2.3.1 Compute shaders for explicit integration

Explicit integration methods are inherently parallelizable because each particle (or grid cell for Eulerian implementations) can be computed independently. Computing each time step is computationally quite simple (Johnson et al, 2024), however explicit methods are only conditionally stable - this means that the time step [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%20t#0) must be small enough to maintain stability. The time step size is traditionally calculated by the Courant-Friedrichs-Lewy (CFL) condition, which changes [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%20t#0) based on the simulation resolution (which can be adaptive) and fluid velocity (Eisemann et al., 2013). Smaller time steps mean more iterations are needed to calculate fluid movement over the same timespan, which increases the computational overhead. Implementing a compute shader that uses explicit numerical integration is usually simpler than using an implicit or hybrid approach, however can suffer from stability issues, especially with larger time steps (Kass & Miller, 1990). In addition, handling constraints such as obstacles and boundaries without introducing instability can be challenging.

2.3.2 Compute shaders for implicit integration

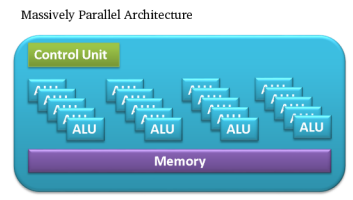
Implicit integration requires solving large linear systems, for example pressure projection in the Navier-Stokes equations. Solving these large linear systems requires inverting matrices, which can be difficult to parallelize and is computationally very expensive (Vantzos et al, 2018). Storing these large matrices at runtime can also create significant memory overhead. There are techniques to achieve the same result without as much computational overhead, such as Conjugate Gradient (CG) and Multigrid, however they are challenging to implement within a parallel environment and are beyond the scope of this paper. Implicit integration with compute shaders is more complex than explicit integration, and ensuring both numerical accuracy and performance are maintained can be difficult (Va et al, 2022). The benefit is that implicit methods can maintain stability with much larger time steps, but the computation required per step is much higher.

2.3.3 Compute shaders for hybrid integration methods

Hybrid approaches to integration combine explicit and implicit techniques, using explicit methods to calculate some terms (such as advection) and implicit methods to calculate others (such as pressure projection). These approaches attempt to strike a balance between stability and performance. The complexity of this kind of implementation is significant, as it requires the use of both previous techniques as well as careful consideration of which terms to treat explicitly and which to treat implicitly. This will depend largely on the desired levels of stability and performance - because there are two different types of solves going on, handling boundaries while keeping the simulation synchronised can be very challenging. This is because different boundary calculations will be required depending on which step of the equation is being solved (and whether the current step is being solved explicitly or implicitly).

GPU based fluid simulations are generally much faster than those running on one or multiple CPU cores. CPUs are responsible for a wide variety of computational tasks, whereas a GPU is mostly dedicated to visual workloads such as rasterization, which require huge amounts of floating-point arithmetic but do not require other elements of the CPU such as large cache capacity and advanced flow control. (Aissa, 2017).

Fluid simulations can be run 25 times faster using only a single GPU than on multicore CPUs (Bailey et al, 2009). In addition to the computational gain, The power efficiency of the process can be improved by more than an order of 10. (Schreiber & Neumann, 2010). In microprocessors, integer math is performed by Arithmetic Logic Units (ALUs). Typically, modern CPUs will have a few ALUs per core, some for integer math and some floating point operations (Kanter, 2010). Modern GPUs often have hundreds or thousands of ALUs, specialised only for floating point math.

Figure 2.0 Massively Parallel Architecture contains many ALUs for floating point math (Aissa, 2017, p.5).

2.4 Common Artifacts in High-Energy Fluid Simulation

2.4.1 Energy Non-Conservation

An inaccurate implementation of the chosen numerical integration method can lead to subtle energy gains or losses over many time steps, which will eventually cause the simulation to no longer appear realistic. Explicit integration methods are more likely to cause unbounded energy gain over time, which can cause the fluid to “blow up” (Pazmino, 2025). If energy is instead lost over time, the fluid will appear unnaturally calm. This can happen if the simulation resolution is too small, as small scale features such as turbulence can be lost. (Madera et al, 2015) Implicit methods are safer against energy non-conservation because they take more than just the current state of the fluid into account.

2.4.2 Excessive damping

Many fluid implementations can cause excessive damping over time by utilizing artificial viscosity to stabilize high energy areas, smoothing over high frequency oscillations (Wang et al, 2025). If this artificial viscosity is excessively applied, natural fluid details such as vortices and turbulence can become too subtle, making the fluid appear too smooth (Nabizadeh, 2024). Treating the diffusion term of the Navier-Stokes equations implicitly frequently causes excessive damping (Huang, 2024).

2.4.3 Exaggerated splashes

If the resolution of a simulation is too low, or numerical errors occur, high energy fluids can exhibit violent interactions such as unrealistically powerful splashes or an excessive number of individual droplets. These are some of the most dramatic artifacts that can occur within fluid simulations, and their presence in a video game environment is likely to reduce the player’s feeling of immersion significantly (Huang, 2024).

2.4.4 Temporal Aliasing

High frequency waves or small, fast moving vortices may not be captured if the time step is too large (Pazmino, 2025). This can lead to jerky and unnatural movement referred to as temporal aliasing. This is usually fixed simply by shrinking the time step, although this comes at a computational cost.

2.4.5 Grid-Aligned Artifacts

Eulerian simulations (those based on a grid of nodes) can exhibit specific artifacts not seen in Lagrangian implementations such as SPH. Artifacts aligned with the grid structure can appear, including unnatural flow patterns or axis-aligned vortices (Koumoutsakos, 2008). These artifacts are most often caused by incorrect discretization of the Navier-Stokes equations into a form that can be solved on a finite grid (Nabizadeh, 2024).

2.5 Evaluating Stability and Realism

2.5.1 Stability

Stability in the context of fluid simulation refers to how well the simulation maintains numerical stability over many time steps. Simulations with high stability are more likely to adhere to fundamental physical laws, without diverging and without the fluid losing or gaining energy overall (Madera, 2015). Unstable simulations are especially vulnerable to catastrophic failures or error accumulation when the conditions within the fluid are extreme, such as having zones of very high pressure or velocity. Stability can be especially difficult to maintain when the fluid is bounded by or otherwise interacting with complex geometry, as it can create areas of differing pressures and velocities.

2.5.2 Realism

Realism refers to how accurately the simulation depicts the physical behaviour of real life fluids, including their interactions with solid obstacles, boundaries, and other fluids with different properties. Realistically simulated fluids should respect the laws of physics, including conservation of energy, momentum and mass over long periods of time. A realistic simulation should result in consistently visually plausible results with a variety of differing factors and starting conditions (Saha, 2025). Natural phenomena, such as vortices, turbulent flows, and splashes, should occur, and artifacts from numerical errors or incorrect time step sizes should be avoided.

2.5.3 Qualitative Benchmarks

Fluid simulation accuracy can be evaluated via comparison to real-world footage of fluids or experimental data (Saha, 2025). Running a simulation for many time steps can reveal artifacts such as blow-up, axis-aligned vortices, and excessive damping. High-energy fluids in real life display many small details, such as turbulence and vortices, that will not be present in simulated fluids experiencing excessive damping.

2.5.4 Quantitative Benchmarks

A simple measurement of the total energy of the fluid over many time steps can reveal energy buildup or damping - a stable, realistic simulation should maintain the same energy level throughout the simulation time. The same measurement can be made for the total mass of the fluid, which should also remain constant. It is also possible to quantify the strength of vortices within the fluid, and comparisons can be made to a higher resolution, baked reference simulation with the same starting conditions and boundaries (Horvath, 2016).

2.5.5 Physical Benchmarks

Reynolds Numbers characterize the balance between viscosity and inertia experienced by fluids. When comparing simulations to baked or real footage, the simulation fluid should have the same Reynolds number as the reference material, as this ensures that the same behaviour is expected (Horvath, 2016). The Froude Number characterizes the balance between inertia and gravity, and can be used in a similar way to compare the size and frequency of waves to reference data.

2.5.6 Performance Benchmarks

Different implementations will use different resources, so to evaluate overall efficiency GPU utilization, frame rate and memory (VRAM) usage should all be considered and compared. For iterative solvers such as CG and Multigrid, the rate of convergence can be measured to evaluate the efficiency of the numerical method (Huang, 2024). Measuring the stability with multiple time step sizes can reveal the maximum time step size where the simulation is still stable. Using the largest time step available will provide the best performance in almost all cases.

2.6 Conclusion

The exploration of numerical methods and computational techniques for real-time, high-energy fluid dynamics reveals a nuanced landscape of trade-offs between realism, stability, and performance. Based on the literature reviewed, implicit integration methods may not be a good fit for the type of real-time interactive simulation required in video games. This is because, despite offering good stability over long periods even with large time steps (Vantzos et al, 2018), they have too high of a computational cost to be implemented practically within a game environment, where performance is of the utmost importance. Similarly, semi-implicit (hybrid) approaches, which allow us to leverage the strengths of both explicit and implicit integration methods, are also deemed impractical due to the significant complexity of implementation. In a fast paced development environment such as a game studio, significant time constraints are likely to be present which a hybrid implementation might struggle to adhere to (Huang 2024). Taking the reviewed literature into account, a simpler method is likely to be more suitable for real-time applications - especially interactive environments, where players may create situations that cannot be predicted in advance.

Explicit integration methods emerge as the most viable choice for real-time fluid dynamics in games, offering great computational efficiency. Although discretization is still challenging, compared to other methods they are simpler to parallelize and implement within a compute shader (Pazmino, 2025). A significant difficulty will be the management of time step sizes to maintain both stability and performance (Liu, 2023). The parallel-friendly nature of explicit integration methods makes them ideal for implementation within a real-time environment, as the GPU can be leveraged efficiently through compute shaders.

Both Eulerian and Lagrangian methods present compelling advantages in terms of spatial discretization. Eulerian (grid-based) methods are used widely in games where large-scale fluids need to be modelled. Examples include the fluid simulations in Half-Life 2 (Valve Corporation, 2004) and Assassin’s Creed 3 (Ubisoft, 2012), which leverage Eulerian methods to achieve natural looking water and smoke effects. Other games, such as PixelJunk Shooter and Wave Race 64, utilize Lagrangian methods such as SPH to produce effective water detail such as splashes and turbulence. A hybrid Eulerian/Lagrangian method would produce a good middle ground but is deemed to have the same shortcomings as a hybrid numerical integration method and therefore is unsuitable for real-time applications such as video games (Koumoutsakos, 2008), unless it is pre-baked using software such as Houdini (SideFX 2025) or Blender (although in this case it ceases to be interactive) (Blender Foundation 2025).

Figure 3.0: Water in Half-Life 2 uses a Eulerian method to simulate water movement when physics objects interact with the water surface (Image: IGN 2015).

3.0 Output Design

3.1 Summary

To demonstrate the practical application of compute shaders for the simulation of high energy fluids in real-time, two differing approaches to fluid simulation will be implemented using HLSL compute shaders in Unity (Unity Technologies 2025). One of the implementations will be based on Eulerian methods, utilising a grid to separate the fluid calculations across the whole space that the fluid occupies. The other will be a Lagrangian implementation, and programming two simulation methods will allow direct comparisons in stability, performance and visual realism between the two.

3.2 Eulerian Simulation

A 3D grid will represent the fluid domain, with compute shaders handling the flow calculations at each node of the grid. The simulation will focus on visually compelling elements such as vortices and plumes, as these will be the most noticeable and immersive to players using real-time applications such as video games (Pazmino, 2025).

3.3 Lagrangian Simulation

A particle based approach, specifically SPH, will be implemented where the fluid is represented as thousands of particles spread throughout the domain. Compute shaders will allow the parallelization of particle calculations which will allow for high resolutions without compromising the real-time nature of the simulation. Explicit integration will be used to solve the SPH formulation of the Navier-Stokes equations.

3.4 Metrics

The simulations will be compared with several benchmarks. The three main points of evaluation are performance, stability, and realism. Performance will be determined through framerate, computation time per time step, and GPU utilization. The stability of the simulations will be tracked over many time steps - throughout the simulation the total mass and energy across the whole fluid domain will be plotted, which allows calculation of the drift in those values over time. Simulations that achieve a smaller drift value are more stable. Realism will be evaluated using visual comparisons between the two implementations as well as comparisons to pre rendered fluid simulations created in animation software Blender (Blender Foundation, 2025) (Blender uses a FLIP based implementation called Mantaflow). Any artifacts affecting the simulations will be noted and considered.

3.5 Expected outcomes

By implementing both Eulerian and Lagrangian simulations, the output design will provide a direct comparison of their suitability for real-time applications through evaluation of their performance, stability and realism. The Eulerian simulation is expected to be a stable and efficient system for simulating large scale fluid behaviours, making it suitable for applications where an environment artist or game designer might need to control the fluid behaviour to achieve an artistic or gameplay effect (such as water slowly filling a submarine). In contrast, the Lagrangian simulation is expected to be superior at simulating small scale, high detail interactions such as vortices, which would make it more suitable for physical effects such as water splashing around in a bottle or smoke coming off of a fire.

This dual implementation will not only demonstrate the advantages of compute shaders for the real-time simulation of physically accurate, high-energy fluid dynamics, but also provide insights into the practical trade-offs between Eulerian and Lagrangian implementations, guiding future work in choosing and optimising an appropriate method for their specific use case.

4.0 References

Adegboye, F., Danjuma, Y. J. & Aniki, S. A. (2023). *Derivation and Implementation of a Linear Multi-Step Method for Numerical Solution of Initial Value Problems of First-Order Ordinary Differential Equations with Zero Stability.* p. 81-93 From the 59th *MAN Annual National Conference,* 2023, ResearchGate. Accessed at https://www.researchgate.net/publication/381806455\_Derivation\_and\_implementation\_of\_a\_Linear\_Multi-step\_method\_for\_numerical\_solution\_of\_initial\_value\_problems\_of\_first-order\_ordinary\_differential\_equations\_with\_zero\_stability on 14/03/2025

Aissa, M. (2017). *GPU-Accelerated CFD Simulations for Turbomachinery Design Optimization*. Accessed at https://www.researchgate.net/publication/320979427\_GPU-accelerated\_CFD\_Simulations\_for\_Turbomachinery\_Design\_Optimization on 03/03/2025

Bailey P., Myre J., Walsh S.D., Lilja D.J., and Saar M.O.. (2009) *Accelerating lattice boltzmann fluid flow simulations using graphics processors*, International Conference on Parallel Processing: Vienna, Austria. Accessed at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5362489 on 13/03/2025

Batchelor, G.K. (2000) *Kinematics of the Flow Field,* in *An Introduction to Fluid Dynamics*. Cambridge: Cambridge University Press (Cambridge Mathematical Library) p. 71-130 Accessed at https://www.cambridge.org/core/books/an-introduction-to-fluid-dynamics/18AA1576B9C579CE25621E80F9266993 on 03/03/2025

Blender Online Community (2025) *Blender* - A 3D Modelling and Rendering Package http://www.blender.org Accessed on 03/03/2025

Borthakur, P. P. & Sarmah, P. (2013). *A Review of Simulation Techniques as an Important Tool for Solving Complex Problems.* From the *International Journal of Scientific and Engineering Research*. p.472-480 ResearchGate. Accessed at https://www.researchgate.net/publication/353156350\_A\_Review\_of\_Simulation\_Techniques\_as\_an\_Important\_Tool\_for\_Solving\_Complex\_Problems on 04/03/25.

Eisemann, E., Assarsson, U. & Schwarz, M. (2013). *Real-Time shadows.* Accessed at https://www.taylorfrancis.com/books/mono/10.1201/b11030/real-time-shadows-elmar-eisemann-ulf-assarsson-michael-wimmer-michael-schwarz on 07/03/2025

Giner, L., Czerny, R., Gruber, C., & Rauscher, F. (2024). *Generic and Automated Drive-by GPU Cache Attacks from the Browser.* p. 128-140 From *ASIA CCS 2024* ACM. Accessed at https://dl.acm.org/doi/pdf/10.1145/3634737.3656283 on 14/03/25

Hirom, K. (2024). *Investigating the Impact of Inclined Plates and Perforated Baffles on Sedimentation Tank Performance through Computational Fluid Dynamics Modelling.* ResearchGate. Accessed at https://www.researchgate.net/publication/388497500\_Investigating\_the\_Impact\_of\_Inclined\_Plates\_and\_Perforated\_Baffles\_on\_Sedimentation\_Tank\_Performance\_through\_Computational\_Fluid\_Dynamics\_Modelling on 02/03/25

Horváth, Z. (2012). *Real-Time Particle Simulation of Fluids*. Accessed at https://www.researchgate.net/publication/266223633\_Real-time\_particle\_simulation\_of\_fluids on 09/03/2025

SideFX (2025) *Houdini - 3D modeling, animation, VFX, look development, lighting and rendering* Accessed at https://www.sidefx.com/ on 14/03/2025

Huang, Y. (2024). *Variational Inference via Smoothed Particle Hydrodynamics*. arXiv. Accessed at https://arxiv.org/pdf/2407.09186 on 10/03/2025

InsPyro, 2019, Figure 1.0: A fluid simulation to virtually test the safety of extraction hood ventilation designs. Accessed at: https://inspyro.be/cfd-for-health-and-safety/ on 14/03/25

Iskhakov, A. S., & Dinh, N. T. (2020). *Physics-integrated machine learning: Embedding a neural network in the Navier-Stokes equations*. arXiv. Accessed at https://arxiv.org/pdf/2008.10509 on 02/03/2025

Johnson, B. A., Bu, S., Mundy, C. J., & Ananth, N. (2024). *MixPI: Mixed-Time Slicing Path Integral Software for Quantized Molecular Dynamics Simulations.* arXiv. Accessed at https://arxiv.org/pdf/2411.11988 on 14/03/2025

Kanter, D. (2010). *Intel’s Sandy Bridge Microarchitecture*, Real World Tech, 25/07/2010 Accessed at: https://www.realworldtech.com/sandy-bridge/ on 13/03/2025

Kass, M. & Miller, G. (1990). *Rapid, stable fluid dynamics for computer graphics.* From *Computer Graphics, volume 24, number 4*. p. 49-57 ACM SIGGRAPH. Accessed at: https://dl.acm.org/doi/pdf/10.1145/97879.97884 on 02/03/2025

Khedkar, K. M. (2018). *A Unified Fluid Flow Solver on Hybrid Unstructured Grids.* ResearchGate. Accessed at https://www.researchgate.net/publication/330884582\_A\_Unified\_Fluid\_Flow\_Solver\_on\_Hybrid\_Unstructured\_Grids on 14/03/25.

Koumoutsakos, P. & Cottet, G. H. (2009). *Flow simulations using particles - Bridging Computer Graphics and CFD*. Accessed at https://www.researchgate.net/publication/29625483\_Flow\_simulations\_using\_particles\_-\_Bridging\_Computer\_Graphics\_and\_CFD on 06/03/2025

Liu B., Wang M., Yang Y., Shao Y. (2023). *Building a Real-Time System on GPUs for Simulation and Rendering of Realistic 3D Liquid in Video Games*. Accessed at https://dl.acm.org/doi/pdf/10.1145/3587423.3595537 on 13/03/2025

Madera, F. A., Moo-Mena, F., & Ayala, E. (2015). *Particle simulation with GPUs shading languages*. From the *International Journal of Computer Science Issues, Volume 12, Issue 5*. P. 183-192 Accessed at: https://www.researchgate.net/publication/352159523\_Particle\_Simulation\_with\_GPUs\_Shading\_Languages on 03/03/2025

Markopoulos, P. (2019). *Simulating an exciting game experience within virtual reality* ResearchGate. Accessed at:https://www.researchgate.net/publication/340983491\_Simulating\_an\_exciting\_game\_experience\_within\_virtual\_reality\_THESIS\_Masters\_of\_Arts\_MA\_in\_Virtual\_Reality on 14/03/25

Mashayekhi, O., Shah, C., Qu, H. & Lim, A. (2018). *Automatically Distributing Eulerian and Hybrid Fluid Simulations in the Cloud*. Accessed at https://dl.acm.org/doi/pdf/10.1145/3173551 on 12/03/2025

Mastin, Gary A.; Watterberg, Peter A.; Mareda, John F. (1987). *Fourier Synthesis of Ocean Scenes*. IEEE Computer Graphics and Applications. 7 (3): 16–23. Accessed from archive at https://web.archive.org/web/20160305101113/http://www-evasion.imag.fr/people/Fabrice.Neyret/images/fluids-nuages/waves/Jonathan/articlesCG/fourier-synthesis-of-ocean-scenes-87.pdf on 09/03/2025

Moraru, A., Bruland, O., Perkis, A. & Rüther, N. (2019). *Visualizing Hydrodynamic Fluid Simulations Within an Immersive Experience as a Scientific Dissemination Strategy.* Accessed at https://www.researchgate.net/publication/335700786\_Visualizing\_hydrodynamic\_fluid\_simulations\_within\_an\_immersive\_experience\_as\_a\_scientific\_dissemination\_strategy on 08/03/2025

Nabizadeh, M. S., Roy-Chowdhury, R., & Yin, H. (2024). *Fluid Implicit Particles on Coadjoint Orbits*. arXiv. Accessed at https://arxiv.org/pdf/2406.01936 on 01/03/2025

Nishidate, Y. & Fujishiro, I. (2024). *Efficient Particle-Based Fluid Surface Reconstruction Using Mesh Shaders and Bidirectional Two-Level Grids*. Accessed at https://dl.acm.org/doi/abs/10.1145/3651285 on 11/03/2025

Pazmino, S. A. (2024). *A Coupled SPH-LSDEM Scheme That Allows for Solid-Fluid Interaction*. ResearchGate. Accessed at https://www.researchgate.net/publication/388177854\_A\_coupled\_SPH-LSDEM\_scheme\_that\_allows\_for\_solid-fluid\_interaction on 08/03/2025

Saha, G. (Preprint)(2025). *Validation Insights on Heat Transfer and Entropy Generation in Enclosures: A Beginner's Guide* Accessed at https://www.researchgate.net/publication/388402847\_Validation\_Insights\_on\_Heat\_Transfer\_and\_Entropy\_Generation\_in\_Enclosures\_A\_Beginners\_Guide on 05/03/2025

Schreiber, M. & Neumann, D. (2010). *GPU-based simulation and visualization of fluids with free surfaces*. Accessed at: https://www.researchgate.net/publication/275522832\_GPU\_based\_simulation\_and\_visualization\_of\_fluids\_with\_free\_surfaces on 01/03/2025

Shaffer, C. A. (1992). *Dissertation Abstracts in Computer Graphics*. Accessed at https://dl.acm.org/doi/pdf/10.1145/142403.988636 on 10/03/2025

Ubisoft (2012) *Assassin’s Creed III*. Accessed at https://www.ubisoft.com/en-gb/game/assassins-creed/assassins-creed-3 on 14/03/2025

Unity Technologies (2025) *Unity Real-Time Development* - https://unity.com/ Accessed at 14/03/2025

Va, H., Choi, M. H., & Hong, M. (2022). *Real-time Volume Preserving Constraints for Volumetric Model on GPU.* From *Computers, Materials and Continua* p. 831-848. ResearchGate. Accessed at https://www.researchgate.net/publication/360685268\_Real-time\_Volume\_Preserving\_Constraints\_for\_Volumetric\_Model\_on\_GPU on 14/03/2025

Valve Corporation (2004) *Half-Life 2* Accessed at https://store.steampowered.com/app/220/HalfLife\_2/ on 14/03/2025

Vantzos, O., Raz, S., & Ben-Chen, M. (2018). Real-time viscous thin films. Accessed at https://dl.acm.org/doi/pdf/10.1145/3272127.3275086 on 04/03/2025

Wang, M. & He, Y. (2016). *A System for Rapid Exploration of Shader Optimization Choices*. Accessed at https://dl.acm.org/doi/pdf/10.1145/2897824.2925923 on 14/03/2025

Wang, F., Sun, Z., & Hu, X. (2024). *An efficient truncation scheme for Eulerian and total Lagrangian SPH methods*. arXiv. Accessed at https://arxiv.org/pdf/2405.05155 on 06/03/2025

Wang, F., Sun, Z., & Hu, X. (2025). *A Weakly Compressible SPH Method for RANS Simulation of Wall-Bounded Turbulent Flows*. arXiv. Accessed at https://arxiv.org/pdf/2501.18397 on 06/03/2025