Fast, Adaptive Methods for Weakly-Compressible Smoothed Particle Hydrodynamics

Submitted in partial fulfilment of the requirements
of the degree of
Doctor of Philosophy
by

Abhinav Muta
Roll No. 153010007

Supervisor:
Prof. Prabhu Ramachandran

Department of Aerospace Engineering
Indian Institute of Technology Bombay

July, 2023

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Abhinav Muta
Roll No. 153010007

Date: 8th June, 2023

Acknowledgements

Back in 2017, Professor Prabhu Ramachandran, my advisor and mentor, put forth the idea to continue my Masters to a Ph.D. Little did I know how this suggestion would profoundly shape my life. My growth as a researcher wouldn't have been possible without professor's continuous support. Thank you, Professor, for motivating me in times of low spirits; for inspiring me with your fecund imagination——I always came out of the meeting charged with curiosity; for tolerating me when I pursued my disparate interests.

To my family, to whom I owe everything; who have been there for me since the very beginning; who, every time I came back home after long periods, made me feel as if I never left. To my father, my first science teacher, for giving me his patience. To my mother, my first teacher, for giving me her humor. To my grandmother, for giving me her strength. Soumya, my beloved, in whom I take much pride in.

One should consider themselves fortunate to be guided by teachers who ignite passion in a subject, an excitement that goes beyond the classroom and continues to last for years after. Aside from the main matter, I found those slight, and often inconsequential——only for the course syllabus——side topics to provoke in me the wish to learn more. I would like to thank professor Prabhu Ramachandran for teaching me Computational Fluid Dynamics, and the importance of computational efficiency and profiling; Professor Debasish Chatterjee for Differential Geometry and Topological methods in data science and Random Processes; Professor Amuthan Ramabathiran for Multi-scale modelling and Continuum mechanics; Professor Aniruddha Sinha for Aeroacoustics. Their teaching bought moments that proliferated over time, much longer after the course, to make the subject beautiful.

I owe much gratitude to Prof. T. N Venkatesh and Dr. Kunal Puri for agreeing to review this thesis and for their insightful comments from the point of view of their respective fields. I wish to thank my thesis committee, Prof. Amit Agrawal and Prof. Kowsik Bodi, for reviewing my

work from its early stages through various reformulations to its current well-developed state. I would also like to thank the several anonymous reviewers who voluntarily reviewed our journal articles.

What a glee a piece of code can be! I cannot imagine my research without the many open-source packages that I had used and still continue to use. I thank all those developers who are giving a creative part of themselves that goes beyond a pay-check.

I am proud to have compassionate friends who put on various roles to get through the challenging times. Dinesh: you are there since the very beginning of my Ph.D., since the day we first installed Linux through the times we customized our Emacs. Thank you for introducing me to the ideas of Nietzsche and many of the open-source packages that I now use. Our many commutes on local trains filled with math discussions will be forever etched in my memory. Kumara: I greatly admire your discipline. Your questions have made me look for answers that immensely increased my understanding. I took pleasure when you picked up to read one of my books on the desk. I always used to look forward to our daily commute back from the lab filled with math, not-so-math and our critical views of all things *system*. Asmelash: thank you for introducing me to the delectable Ethiopian cuisine, and for sharing with me many coffees. Your humor would always bring a smile in me. Pawan: thank you for introducing me to ultimate Frisbee. Our many discussions on the fundamentals of SPH had always been fruitful. I derived great fun with Dinesh, Kumara, Asmelash, and Pawan by being part of the scrabble club, where we all became fierce competitors; the math club, where all became teachers; our coffee outings, where we all became advisors and motivators. Navaneet: thank you for being a kind person and for sharing many things that I am are quite unaware of.

I express my gratitude to my friends Aradhana and Aishwarya, who have been there for me throughout my highs and lows. Aradhana: you have been a great friend to me since the moment I have met you. With your being, you limned a gentler world. Aishwarya: you are a kindred spirit to me. Many a time with your prompts, you have made me reminisce the times gone by, and your belief in my writing has often been a source of encouragement.

Thanks to the Department of IDC for allowing me to take up pottery and weaving. These courses made me look at things from a different perspective, different from the structured way of thinking that I was bought up on. I met passionate people, Taniya Vaidya and Maruthi Rao, who taught me many basics of the fine arts. I would also like to thank, Mr. Kanta Kannaujiya.

My sincere thanks to the Department of Aerospace Engineering for giving me the access to the Aerospace Computational Engine, the high performance computing facility, for the simula-

tions that I have done throughout my research will be made impossible without it.

The beautiful campus of IIT Bombay has often soothed me with its liana of moss warped trees, its verdant grounds, the Powai lake, and the birds during the day, and during the night with the views of the skyline and the Ghats from the roofs of the tallest buildings. I had spent many hours a day, every day, around the lakeside, the cafés, and the beautiful gardens, reading a book or working on a problem or simply walking, the time post-pandemic wouldn't have been as rich without the campus premises.

In the end, it is the city of Mumbai. Many things have changed after the pandemic, and in this transition, while I was groping for certainty, it was the city that I took the most from. It is an energy field crisscrossed with memories that had filled my heart with a deepening sense of sorts. The trains sending pinging sounds through the tracks has held me in place feeling ever so frightened but captivating for what's to come; the thrumming of the markets of Dadar and Kalbadevi has endured me through change; the contrast of the scenes: the rich and the poor, the chawls against the skyscraper has mimicked the vicissitudes of life; the inky blue skies over the curving skyline at Marine Drive made me feel electrified; it's many cultures; old Irani cafés; the art and the architecture; the kind and beautiful people has expanded my horizons, made me strong, and filled me with hope. It is to Mumbai that I dedicate this thesis.

To Mumbai

Abstract

Smoothed Particle Hydrodynamics (SPH) is a meshless, Lagrangian method that can handle a large class of applications with complex physics, such as free-surface flows, multiphase flows, astrophysical problems, and fluid-structure interaction, to name a few. The general nature of SPH makes it a viable method for simulating problems with multiple scales, making it valuable for practical applications in industries. While all this is promising, the standard SPH method using weakly-compressible formulation is computationally expensive making it inefficient over other mesh-based methods. Moreover, significant efforts in adaptive spatial resolution have only recently started appearing in the literature and there is a need to address performance while retaining accuracy.

Fast algorithms are much sought after in every numerical method. Often seen as a fast alternative to the weakly-compressible SPH schemes, incompressible SPH (ISPH) methods sidestep the speed of sound by solving a coupled pressure Poisson equation to model incompressible fluids. The ISPH method constructs a large matrix that is complex to implement on GPUs because of the memory constraints inherent to them. In this work, we examine a matrix-free, simple iterative ISPH (SISPH) that uses an iterative formulation to set up and solve the pressure-Poisson equation, which can be easily implemented on GPUs and CPUs. We propose a novel approach to ensure homogeneous particle distributions and improved boundary conditions. The method is fast and runs on GPUs without requiring complex integration with sparse linear solvers. We show that this approach is sufficiently accurate and efficient compared to other approaches.

Inspired by recent strategies, with the re-emergence of dual-time stepping, the current research proposes and develops a fast scheme for simulating weakly-compressible fluid flows. The dual-time SPH (DTSPH) scheme combines the dual-time stepping with entropically damped artificial compressibility (EDAC) formulation. EDAC formulation offers superior pressure distribution by diffusing high-frequency pressure modes in the incompressible regime. DTSPH forms

the second fast scheme in this work, along with SISPH. These approaches model truly incompressible and weakly-compressible formulations of the Navier-Stokes equation, respectively.

In the adaptive spatial resolution front, we propose an accurate and computationally efficient method for incorporation into weakly-compressible SPH schemes. Particles are adaptively split and merged accurately. Critically, the method ensures that the number of neighbors of each particle is optimal, leading to an efficient algorithm. The algorithm can handle geometry-based adaptive resolution, where the resolution is a function of distance to a solid body, or solution-based adaptive resolution, where the resolution is a function of the computed solution.

Among the contributions is the achievement of a high variation in scales. This allows us to simulate problems using particles having length variations of the order of 1:250 with much fewer particles than currently reported with other techniques. The method is designed to adapt automatically when any solid bodies move. The algorithms employed are entirely parallel.

Finally, we incorporate adaptive resolution methods with a fast DTSPH scheme and show a performance improvement of up to 4.5 times over the standard EDAC formulation maintaining similar accuracy. We provide an open-source implementation of both SISPH and DTSPH schemes and the adaptive particle resolution algorithm. We consider an extensive suite of test problems that cover a wide range of incompressible fluid flow scenarios. All the test problems are fully-reproducible and can be implemented on multi-core CPUs and GPUs.

Keywords: SPH, weakly-compressible SPH, incompressible, solution adaptivity, ISPH, GPU, iterative, dual-time stepping, incompressible fluid flow, variable spatial resolution, adaptivity, moving geometries, open-source code

Contents

List of Figures

List of Tables

1 Introduction

Perhaps it is true to say that one has found from experiences that deeper insight into the basic concepts of a theory comes most often, not from a frontal attack on those concepts, but rather from working upward into the theory itself.

—— Robert Geroch, *General relativity from A to B (1978)*

Computational modeling is an essential tool to model physical phenomenon. Especially true in fluid dynamics, where the mathematical formulations of the governing equations has been established more than a century ago, it is only recently that computational fluid dynamics is shedding light on the many problems where analytical tools are of limited help. As such, there is a need for methods with high reliability, accuracy, and computational efficiency, which are often hard to find in one single numerical scheme.

Traditional numerical schemes have an underlying framework of a mesh which enables the methods to compute derivatives, integrals, and imposition of boundary conditions. However, if one wants to expand the method to solve a general problem with challenging mesh generation and adaptation, the numerical schemes breakdown and significant effort have to be undertaken to overcome the difficulties. Although not relatively new, the meshless methods originate in the works of vortex methods developed in the 60s and 70s. Because of the need for complex simulations, a general class of meshless methods has been pursued.

The Smoothed Particle Hydrodynamics (SPH) method was initially proposed to simulate astrophysical problems by Lucy (1977) and Gingold et al. (1977). It is a mesh-free, Lagrangian, particle-based method that has since been used to simulate incompressible fluids. The weakly-compressible SPH (WCSPH) scheme (Monaghan 1994) was proposed to simulate incompress-

ible flows by treating the fluid as weakly-compressible and using a stiff equation of state. This method has been used to simulate fluid flows with free-surfaces, mixing problems, multiple bodies, gas dynamics problems, and many others. One disadvantage of the scheme is the computational effort per time-step. A bane for problems with various scales, using constant resolution is often prohibitively expensive. A single user with limited computational resources cannot even fathom performing multiple scale simulations even with a simple setup, like a flow past cylinder at a moderate Reynolds number of 9500.

In this work, we develop fast and adaptive numerical techniques to model incompressible fluid flows with multiple, complex, moving bodies. The work can be divided into two parts. The first part involves developing fast methods: reducing the wall-clock time for a given simulation by developing methods that use a more significant time step size. These methods contrast the weakly-compressible methods, where the artificial speed of sound poses severe time-step restrictions. To remedy this, there are other alternate formalisms, like, (i) solving for pressure by iterative methods, which may be complex to set up, (ii) using the iterative approach of Chorin's artificial compressibility, which bypasses the time step restriction by solving the Navier-Stokes equations iteratively in the weakly-compressible limit.

The second consists of developing adaptive-refinement strategies and incorporating them into the solver. Adaptive refinement here means the particles are allowed to split and merge in regions of interest, all the while conserving mass. This refinement is crucial in capturing the desired details without resorting to a computationally intensive method of using a single fine resolution of particles in the whole domain. Adaptive particle refinement is crucial in simulations where there is a need for both finer resolution to essential model features and coarse resolution in the regions which lack the features.

1.1 Incompressible SPH Methods

An incompressible material has the restriction to undergo only isochoric motion. This restriction can be shown to be equivalent to the condition of divergence-free velocity (Chorin et al. 1990). There are many ways to enforce this condition. In SPH, there are two prominent ways to model such materials. One is the truly-incompressible approach, usually referred to as incompressible methods, where the pressure is obtained by solving a pressure-Poisson equation (PPE), imposing the divergence-free condition on the velocity. This class of schemes is called the truly-

incompressible schemes. Another class of schemes to model an incompressible material is the weakly-compressible model. Here, slight perturbations in the density are tolerated. The pressure is evaluated based on a stiff state equation relating pressure to density via a parameter called artificial speed of sound. This model approaches the truly-incompressible model in the limit when the artificial speed of sound tends to infinity. Usually, in SPH artificial speed of sound is taken to be around ten times that of the maximum velocity achievable in the simulation. In this work, we developed two schemes, one that falls under the class of truly-incompressible methods and the other under weakly-compressible methods.

The truly-incompressible SPH (ISPH) schemes have their origin in the work of Cummins et al. (1999) who proposed a projection method where a pressure-Poisson equation (PPE) is solved to ensure a divergence free velocity field. In this, the Laplacian of the pressure is related to the divergence of the velocity field. Shao et al. (2003) proposed a slightly different variant which relates the Laplacian of the pressure to a change in density. Later, Hu et al. (2007) proposed a method that combines both of these approaches to enforce incompressibility. The significant advantage with the class of ISPH schemes is that they do not involve the sound speed and hence can use time-steps that are an order of magnitude larger than the weakly-compressible SPH (WCSPH) schemes.

The ISPH schemes have the disadvantage that they require the solution to large (but sparse) linear systems. This makes implementing them in parallel and on GPUs fairly difficult. The WCSPH implementations are generally more popular as they are much easier to implement and parallelize. The recent work of Chow et al. (2018) discusses many of the challenges in implementing traditional ISPH schemes for large scale computing on GPUs.

An explicit version of the ISPH was first proposed by Hosseini et al. (2007) which focuses on simulating non-Newtonian flows. This approach removes the requirement to solve a linear system and instead sets up an explicit (non-iterative) equation to compute the pressure. A similar explicit approach was used by Rafiee et al. (2009) for solving fluid-structure interaction problems. Barcarolo (Barcarolo 2013; Barcarolo et al. 2014b) validates and explores the issues with Explicit Incompressible SPH (EISPH) and compared the scheme with the WCSPH in the context of internal and free surface flows. Nomeritae et al. (2016) assess the performance of EISPH with a comparison with WCSPH and δ-SPH schemes (Marrone et al. 2011). Basser et al. (2017) simulate multi-fluids with porous media using EISPH and compared their results with experimental data. While Hosseini et al. (2007), Rafiee et al. (2009), and Barcarolo et al. (2014b) solve for the pressure by satisfying a constant density condition (Shao et al. 2003), Nomeritae

et al. (2016) and Basser et al. (2017) use the divergence free condition (Cummins et al. 1999) to obtain the pressure. It is important to note that in all these cases, the authors perform a single step to solve the PPE and do not perform any iterations thus making the method a truly explicit ISPH method.

In the graphics community, the Implicit ISPH (IISPH) scheme (Ihmsen et al. 2014) has been developed that provides an iterative solution to the ISPH formulation. This formulation is tied to the way in which the pressure forces are approximated between pairs of particles. The method does not work well with negative pressures. Our own implementation of this scheme demonstrates some sensitivity to changes in the time-step or choice of smoothing kernels, and it can be more involved to implement than many of the traditional WCSPH-based schemes. The method is however, matrix free, and it has been shown to be close to an order of magnitude faster than traditional schemes.

The original weakly-compressible SPH scheme (WCSPH) was proposed by Monaghan (1994). This scheme treats the fluid as weakly compressible with an artificial sound speed and a stiff equation of state. This allows the scheme to utilize a hyperbolic system of equations and integrate them in time. There are many significant variants of this scheme including a Transport Velocity Formulation (TVF) (Adami et al. 2013) which introduces a transport velocity to ensure particle homogeneity. Similarly, an Entropically Damped Artificial Compressibility SPH scheme (EDAC-SPH)(Ramachandran et al. 2019) has been proposed which introduces entropy by diffusing the pressure. This approach is quite similar to the δ-SPH scheme (Antuono et al. 2010) and both schemes produce superior pressure distributions.

A Predictive-Corrective ISPH (PCISPH) (Solenthaler et al. 2009) has been proposed for use in the graphics community for rapid simulation of incompressible fluids.

Fatehi et al. (2019) proposed density-based dual-time stepping schemes for the SPH method. They propose two different formulations in order to update the pressure in pseudo-time. They perform an accurate discretization of the derivatives. Zhang et al. (2020) on the other hand propose a dual-criteria time-stepping strategy that is quite different from the method proposed in (Fatehi et al. 2019; Rouzbahani et al. 2017). They use an “acoustic time-step” to relax the pressure and an “advective time-step” to update the verlet lists used for neighbor computation.

These variations improve the traditional WCSPH scheme in various aspects: particle homogenization, accurate pressure distribution, and performance.

1.2 Adaptive Resolution in SPH

In the context of incompressible and weakly-compressible fluid flow, there have been some valuable developments starting with the pioneering work of Feldman et al. (2007) where the particles are adaptively split in an accurate manner. This work has been extended further to include particle merging by Vacondio et al. (2013b, 2016) and applied to the shallow water equations (Vacondio et al. 2012b), soil simulation (Reyes López et al. 2013), fluid-structure interaction (Hu et al. 2019). The method has been designed to be very accurate and a great deal of care is taken when splitting and merging particles.

Barcarolo et al. (2014a) and Sun et al. (2017a), and Chiron et al. (2018) refine each coarse particle (also called parent particle), in two dimensions, into 4 child particles but also retain the coarse particle. The parent particles are passively advected in the refined regions. This implies that each coarse particle effectively splits into five particles. This reduces the number of refined particles when compared with Vacondio et al. (2013b) who splits into 7. The significant advantage with this approach is that de-refining particles is simple to implement; the parent particles are re-activated and the child particles are removed. This approach has also been used for some impressive multi-resolution simulations using the δ^+-SPH scheme (Sun et al. 2017a, 2018). Another significant advantage is that the smoothing length chosen is much smaller than the typical values chosen in the approaches of (Vacondio et al. 2013b). In order to handle the interactions between the child and parent particles, a particle property γ, is added to each particle. In the transition regions this value is between 0 and 1 whereas in regions with uniform particle smoothing length, the value is either 0 or 1. When the value is zero for a particle, the particle is effectively switched off and when it is one it is active. Intermediate values allow for the use of both the parent and child particles.

Chiron et al. (2018) further refined this method by taking inspiration from traditional Adaptive Mesh Refinement techniques to create an Adaptive Particle Refinement (APR) procedure. In the intermediate regions where larger particles are refined into smaller particles, both the parent and child particles are retained and only particles of the same size interact and the properties are carefully interpolated between the parent and child particles.

Recently, another approach for dynamic particle splitting and merging has been proposed by Yang et al. (2017, 2019) and applied to multiphase fluid simulations. This approach is similar to that employed by Vacondio et al. (2013b) but each coarse particle is only split into two child particles. The parent particle is removed. However, the placement of the child particles

is done carefully along the perpendicular bisector of the line joining the parent particle to its nearest particle. This method will only work in two dimensions and no procedure for the three-dimensional case is proposed. The advantage with this approach is that the particle refinement is much more gradual without a very large increase in the number of particles. Merging is done only between two particles and therefore there is no profusion of particles. The proposed method also elegantly handles gradual refinement of the resolution around an interface using a single parameter. This has been demonstrated for multiphase problems (Yang et al. 2019). An alternative to the distance-based spatial adaptation is the recovery-based a *posteriori* error estimator (Hu et al. 2019), where the error in the SPH velocity gradient is measured and particles are adaptively refined in regions where the error exceeds a tolerance.

In the area of computer graphics, Desbrun et al. (1999) use splitting and merging operators in the SPH method and applied it to highly deformable structures. Adams et al. (2007) use extended local feature size to adaptively refine the particles in the regions of geometric interest. Solenthaler et al. (2011) use two-scales, a lower resolution and a higher resolution, and couple the two with appropriate boundary conditions and feedback forces. However, these works are designed more for computer graphics applications and do not test the accuracy with any standard benchmark problems.

Spreng et al. (2014) have proposed the use of the method of Vacondio et al. (2013b) for performing adaptive particle resolution for structural mechanics problems. They also propose a novel method to merge multiple particles by considering neighboring particles which are identified in two different ways. In a more recent work, Spreng et al. (2020) have proposed the use of adaptive refinement to improve the discretization errors.

Recently, Sun et al. (2021) have employed the adaptive particle refinement and de-refinement approach to strongly compressible, multiphase fluid flows. However, the main focus of the adaptive particle refinement is to ensure a homogeneous and isotropic particle distribution when the fluid is highly compressible.

1.3 Motivation

The first part of this study introduces the Simple iterative Incompressible SPH (SISPH) scheme. The SISPH scheme emerges as a promising solution to address the limitations encountered by existing methods. In a study conducted by Barcarolo et al. (2014b), the authors explore

internal flows and flow past a circular cylinder, they do so for low Reynolds numbers where the chances of particle disorder are less. In fact, in (Barcarolo 2013), it is found that there is some particle voiding present for the case of a lid driven cavity at a Reynolds number of 3200. Moreover, in EISPH, the PPE is solved using a single step. This approach does not work well for high Reynolds number cases and we demonstrate these issues in the current work. Previous works employing the EISPH do not explore the performance on GPUs where the scheme has a high potential for rapid, accurate simulation. Furthermore, they do not simulate the Taylor-Green vortex which typically fails to work unless a careful implementation of particle shifting is included.

Both the WCSPH and ISPH schemes suffer from inaccuracies when the particles become disordered. In flows with significant shear, the particles can become significantly disordered leading to poor accuracy and particle clumping in extreme cases. The ISPH community has developed a few particle shifting techniques (Lind et al. 2012; Skillen et al. 2013; Xu et al. 2009) that add a small amount of particle motion to ensure uniform distributions. These improve the particle distribution at the cost of a negligible amount of diffusion. It does requires some tuning to handle free surfaces.

For the WCSPH schemes, the Transport Velocity Formulation (TVF) (Adami et al. 2013) and the Generalized TVF (Zhang et al. 2017) provide a slightly different and more accurate way of generating homogeneous particle distributions. The method relies on the fact that the naive SPH gradient of a constant background pressure field will generate forces that tend to push particles into a uniform configuration. Hence, the particles are moved with a "transport velocity" which is the combination of the fluid velocity and the perturbation velocity, restoring the uniformity. The background pressure is subtracted from the main momentum equation through an artificial stress term. The Generalized TVF (Zhang et al. 2017), extends the TVF to free-surface flows (Adepu et al. 2021). But, this has not been explored to work with the ISPH scheme.

In summary, the motivation behind the SISPH scheme is to address the limitations of existing schemes and incorporate the Generalized TVF to handle free-surface flows. By doing so, it aims to improve accuracy, minimize the impact of particle disorder, and extend the capabilities of the ISPH scheme.

In the subsequent chapter, we present the dual-Time SPH (DTSPH) scheme. The DTSPH scheme is developed to tackle the severe time step limitations of the weakly-compressible schemes. The original WCSPH and their derivatives generally suffer from a large amount of pressure oscillations and the δ-SPH scheme (Antuono et al. 2010; Marrone et al. 2011) reduces

these oscillations by introducing a dissipation into the continuity equation. All of these schemes employ an artificial sound speed and this places severe time step limitations due to stability considerations.

The truly-incompressible SPH (ISPH) schemes satisfy incompressibility by solving a pressure-Poisson equation. These approaches eliminate the need for evolving the pressure at the sound speed and this significantly increases the allowed time steps. The difficulty with the projection and incompressible schemes is the requirement to solve a linear system of equations which can be time consuming and involved.

Recently, Zhang et al. (2020) proposed the dual-criteria time-stepping to improve the computational performance of the WCPSH method. In this methods the authors propose an advection time-step that is largeer than the acoustic time-step. The neighbours lists are only updated every step of the advection time-step. The benefits of optimal neighbor computation results in a performance improvement without significant complexity albeit at the cost of increased memory. Our proposed DTSPH scheme differs significantly from this approach.

Finally, the last segment of this thesis focuses on the development of adaptive particle refinement in SPH. It is important to note that the aforementioned works described in the preceding section suffer from various limitations. The accuracy of (Vacondio et al. 2013b) comes at a significant cost since each coarse particle splits into 7 particles in two dimensions and around 14 in three dimensions. This leads to an enormous increase in the number of particles as the regions are refined. The particle de-refining method merges particles pair-wise and it is argued (Chiron et al. 2018) that the method is computationally expensive since the rate of splitting particles is significantly larger than the rate of merging. While the method is designed to be accurate, the resulting refined particles also employ a very large smoothing radius in comparison to what would be expected in a fixed particle size discretization of the problem with a similar number of particles. This poses significant additional performance limitations on the method. Moreover, the method relies on manual specification of the spatial regions where the adaptation is desired. This is inconvenient in general and especially when the bodies are moving.

The difficulty with the approaches of Barcarolo et al. (2014a) and Sun et al. (2017a), and Chiron et al. (2018) is that coarse particles effectively split into five particles in each level of refinement. Furthermore, there are additional complications due to the special handling required for the parent and child particles either by the use of the γ parameter or by the use of prolongation and restriction operations in the APR method. It is also not entirely clear what would happen in high-strain fluid flows where the four child particles would drift significantly apart away from

the parent particle. It is not clear if the proposed algorithm of Spreng et al. (2014) is parallel as the details of the implementation are not discussed.

In the work of Yang et al. (2017) it appears that no detailed study of the accuracy of the method has been performed. However, previous accuracy studies by (Feldman et al. 2007) suggest that splitting particles into only two child particles would introduce significant error into the solution. Moreover, the method has only been demonstrated for two-dimensional fluid flows.

SPH is increasingly being utilized in various practical areas such as mixing agents into fluids (Robinson et al. 2008), particle suspensions (Peng et al. 2021), landslides (Tan et al. 2017), river ice jams (Robb et al. 2016), and flooding simulations (Liu et al. 2022). In particular, given the current climate scenario, flooding simulations could prove to be crucial in places like Mumbai.

To conduct these simulations effectively, a robust high performance computational framework is required. In our work, we develop fast numerical algorithms, and efficient adaptive particle refinement techniques, which are crucial for conducting large-scale simulations. This could potentially establish SPH as the preferred numerical method for simulations in these areas. We are also driven to develop tools and offer them as open-source packages, allowing researchers to incorporate them in their own work.

1.4 Objectives of the Current Work

The current work aims to improve the state-of-the-art of SPH schemes in multiple fronts. To start with, we develop two new schemes formulated by careful derivation and add improvements in performance, pressure distribution, and particle distribution. The first one is the simple iterative incompressible SPH (SISPH) which is based on the incompressible SPH formulation, the key goals of this scheme are:

- Propose an incompressible method that is easy to implement, accurate, matrix-free, fast, and works on GPUs.
- Ensure uniform particle distribution by using a transport velocity formulation. Improve the solid wall boundary condition to account for larger time-steps of the incompressible scheme.
- Demonstrate the scheme's capabilities with standard benchmarks: internal flows, external free-surface flows, wind-tunnel type problem requiring an inlet and an outlet.

- Establish the performance improvement of the scheme by comparing with the performance of the traditional matrix-based ISPH scheme.
- Show evidence of the performance improvement achieved on different GPUs as well as multi-core CPUs.

The second scheme we develop is the dual-time SPH (DTSPH) which is closely related to the standard weakly-compressible SPH (WCSPH) formulation. The key ideas are:

- Implement an efficient scheme that employs the dual-time stepping (Chorin 1967) for simulating incompressible flows, and derive the formulation and analyze approaches to improve the efficiency of the scheme.
- Develop a sufficiently general formulation general enough to be used with a variety of different SPH discretizations.
- Demonstrate the performance by comparing with the standard weakly-compressible schemes, like, WCSPH, δ-SPH, and EDAC methods.
- Simulate a variety of benchmark problems in two and three dimensions to showcase the robustness of the scheme.

Subsequently, we propose a novel adaptive particle refinement method for the weakly-compressible SPH method. This work constitutes the following objectives:

- An algorithm to support automatic geometry-based and solution-based adaptivity.
- Demonstrate the accuracy of the algorithm by showing errors in function and gradient approximation, and compare with other the algorithm proposed by Vacondio et al. (2012a).
- Formulate a parallel adaptive algorithm that uses optimum number of neighbors per particle.
- Perform simulations with a large variation in resolution, and using fewer number of particles than the ones reported in the literature.
- Provide an open-source implementation and a reproducible suite of problems.

Finally, we integrate the fast methods with the adaptive particle resolution technique. We demonstrate the performance of the new scheme and compare the accuracy by solving standard problems.

1.5 Reproducible Research

Many researchers (Boettiger 2015; LeVeque 2009; Mesnard et al. 2017) have been promoting reproducible research work to make research easily accessible. Reproducibility makes it easy for other researchers to continue the research without having to rewrite new or debug existing frameworks, which is a highly laborious task (Barba 2016; Peng 2011). In order to facilitate reproducible research, this entire work is made completely reproducible and every figure in this work is automatically generated using the `automan` framework (Ramachandran 2018). The source code for the frameworks described in this work and their respective test cases can be found at the individual links mentioned in the upcoming chapters. The individual repositories can also be found under https://gitlab.com/pypr.

1.6 Organization of the Thesis

The thesis is structured in 7 chapters. Chapter 2 provides the background concepts of the Smoothed Particle Hydrodynamics. Considering formulations relevant to the thesis, after the governing fluid dynamical equations are discussed, we review the weakly-compressible and entropically damped artificial compressible (EDAC) formulations, which form the core of chapters 4 to 6. Note that we do not review the truly-incompressible formulation and its SPH discretization (ISPH) in this chapter, as it is convenient to do so in chapter 3. We also describe two different particle shifting methods, and the boundary condition implementations in this chapter.

Chapter 3 through chapter 6 form the core of the research work, they form the fast and adaptive frameworks of the thesis. In chapter 3 we discuss the fast scheme, simple iterative ISPH, abbreviated as SISPH. The pressure-Poisson equation is set up in a matrix-free form and the convergence of the algorithm is discussed. Several benchmarks are presented to establish the numerical accuracy and show the performance improvement over other schemes. Chapter 3 is a part of published work by Muta et al. (2020a).

Chapter 4 develops the second fast scheme described in this research. It is called dual-time SPH (DTSPH), which is based on the weakly-compressible formulation. We study the scheme's parameters with the analytical solution of the Taylor-Green vortex. After establishing the parameters, we proceed to solve benchmark problems and compare the performance of DTSPH with other schemes. Chapter 4 is part of published work by Ramachandran et al. (2021a).

Chapter 5 develops the adaptive particle refinement with SPH and demonstrates its capabilities using wide range of benchmark problems. Here, we use the weakly-compressible EDAC SPH scheme discretize the PDEs. We demonstrate many practical aspects of adaptive particle refinement such as, solution-adaptivity, moving and complex geometries, and computational efficiency. Chapter 5 is part of two papers by Muta et al. (2022) and Haftu et al. (2022).

Chapter 6 combines the efforts of its preceding chapters chapter 4 and chapter 5: to develop DTSPH with adaptive particle refinement. We demonstrate the capabilities of this Adaptive-DTSPH scheme using the flow past circular cylinder and flow past C-shape problems, and note its superior performance improvement over the EDAC scheme. Chapter 7 presents the contributions, limitations of the work, and future research directions.

2 Smoothed Particle Hydrodynamics

The beginner... should not be discouraged if ... he finds that he does not have the prerequisites for reading the prerequisites. — P.R Halmos
— *(from,* Reed and Simon*, Methods of modern physics volume I)*

2.1 Function Approximation in SPH

In this chapter we briefly introduce the SPH method. We refer the reader to Violeau (2012) for a detailed discussion. In SPH, the particles represent the underlying continuum. A field variable $f(\mathbf{r})$ over a domain $\Omega \subset \mathbb{R}^d$, where d is the dimension, can be approximated by a convolution with the kernel function $W(\mathbf{r}, h)$:

$$f(\mathbf{r}) = \int_{\Omega} f(\mathbf{s})\, W(\mathbf{r} - \mathbf{s},\ h)\, \mathrm{d}\mathbf{s}, \tag{2.1}$$

where h is the smoothing length and $\mathrm{d}\mathbf{s}$ is the differential volume element. The kernel function approximates the delta function in the limit $h \to 0$. The kernel is a compact function such that $W(\mathbf{r}) = 0$ when $|\mathbf{r}| > kh$ for some constant k. In this work, we use the quintic spline kernel in all our simulations, the quintic spline kernel is given by,

$$W(q) = \begin{cases} \sigma_2[(3-q)^5 - 6(2-q)^5 + 15(1-q)^5] & \text{if } 0 \le q < 1, \\ \sigma_2[(3-q)^5 - 6(2-q)^5] & \text{if } 1 \le q < 2, \\ \sigma_2(3-q)^5 & \text{if } 2 \le q < 3, \\ 0 & \text{if } q \ge 3, \end{cases} \tag{2.2}$$

where, $\sigma_2 = 7/(478\pi h(\mathbf{r})^2)$, and $q = |\mathbf{r}|/h$. The kernel is also radially symmetric. The kernel is often written as a function of single variable q, instead of a function of two variables ($\mathbf{r}$, h). The plot of the kernel function and its derivative as a function of r/h is shown in fig. 2.1.

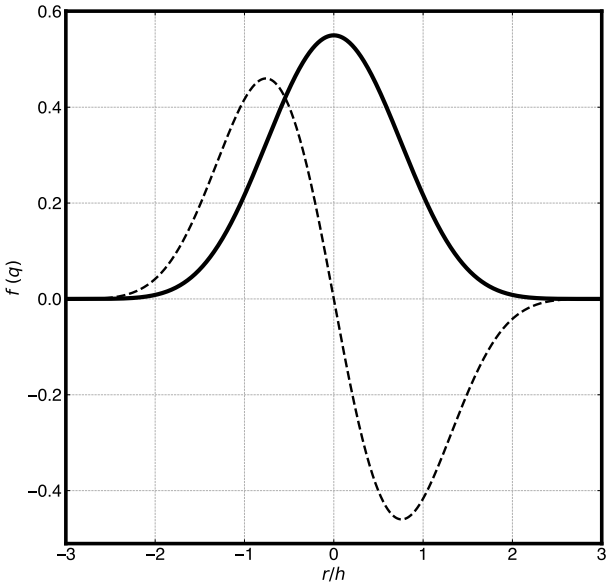


Figure 2.1: The quintic spline kernel and its derivative (dashed line).

To obtain the SPH (discrete) counterpart, the field is discretized into particles having mass m_i, density ρ_i, and the differential volume is replaced by the volume $V_i = m_i/\rho_i$ where the subscript denotes the index of the i^{th} particle at position $\mathbf{r}_i$. The integral is then approximated by summation over all the neighbors in the support of the kernel to give the discrete approximation:

$$f(\mathbf{r}_i) = \sum_j V_j f(\mathbf{r}_j) W(\mathbf{r}_i - \mathbf{r}_j, h). \tag{2.3}$$

If the function f is replaced by density ρ we get the summation density formula in SPH,

$$\rho(\mathbf{r}_i) = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h). \tag{2.4}$$

By differentiating the continuous interpolation eq. (2.1) we arrive at the integral approximation of differential of $f(\mathbf{r})$,

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{r}}(\mathbf{r}) = \int_\Omega f(\mathbf{s}) \frac{\mathrm{d}W}{\mathrm{d}\mathbf{r}}(\mathbf{r} - \mathbf{s}, h)\, \mathrm{d}\mathbf{s}, \tag{2.5}$$

where, the integral is transferred on to the continuous kernel. Using, the above differential approximation, we can obtain the SPH approximation of the derivative as,

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{r}}(\mathbf{r}_i) = \sum_j V_j \; f(\mathbf{r}_j) \frac{\mathrm{d}W}{\mathrm{d}\mathbf{r}}(\mathbf{r}_i - \mathbf{r}_j, h). \tag{2.6}$$

The gradient operator is given as,

$$\nabla_i f(\mathbf{r}_i) = \sum_j V_j \; f(\mathbf{r}_j) \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{2.7}$$

where ∇_i is the gradient w.r.t. $\mathbf{r}_i$. The divergence operator, operating on a vector field $\mathbf{u}(\mathbf{r_i})$, can be approximated as,

$$\nabla_i \cdot \mathbf{u}(\mathbf{r}_i) = \sum_j V_j \; \mathbf{u}(\mathbf{r}_j) \cdot \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h). \tag{2.8}$$

Other forms for the gradient approximation can be derived by using the properties of the kernel function, such as,

$$\int_\Omega \nabla W(\mathbf{r} - \mathbf{s}, h) \, \mathrm{d}\mathbf{s} = 0, \tag{2.9}$$

and,

$$\nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h) = -\nabla_j W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{2.10}$$

which results in a form,

$$\nabla_i \cdot \mathbf{u}(\mathbf{r}_i) = \sum_j V_j \; (\mathbf{u}(\mathbf{r}_i) - \mathbf{u}(\mathbf{r}_j)) \cdot \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{2.11}$$

that satisfy the property that the gradient of a constant function vanishes. Improvements can also be made to ensure linear consistency (Bonet et al. 1999).

2.2 Governing Equations and Formulations

The governing equations of motion for a compressible viscous fluid are the Navier-Stokes equations. The Navier-Stokes equations express the conservation of mass and conservation of momentum, usually accompanied with an additional state equation relating pressure, density, and temperature. In a Lagrangian frame of reference the conservation of mass equation is given by,

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho \nabla \cdot \mathbf{u}, \tag{2.12}$$

where, the Lagrangian/material/convective time-derivative of a quantity f can be written in Eulerian/spatial description as,

$$\frac{\mathrm{d}f(\mathbf{r},t)}{\mathrm{d}t} = \frac{\partial f(\mathbf{r},t)}{\partial t} + \mathbf{u}\cdot\nabla f(\mathbf{r},t), \tag{2.13}$$

where, $\mathbf{r}$, $\mathbf{u}$, ρ, and t denotes the position, velocity, density, and time respectively. The conservation of momentum equation for a single fluid, with a constant viscosity ν and negligible variations in temperature, is written as,

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{u} + \mathbf{f}_{\text{body}}, \tag{2.14}$$

where, p is the pressure, ν is the kinematic viscosity of the fluid, and $\mathbf{f}_{\text{body}}$ is the body force. We need an other equation to complete the above mass and momentum equations. In order to complete the equations (we have five unknowns and four equations) we require an additional relation relating pressure and density. For the incompressible flow modeling there are two approaches that we follow in this work (i) the weakly-compressible formulation and (ii) the entropically damped artificial compressibility formulation.

2.2.1 Weakly-Compressible Formulation

The weakly-compressible formulation is widely used in SPH (Monaghan 2005), where the pressure is recovered from the isothermal Tait-Murnaghan state equation, given by,

$$p = \frac{\rho_0 c_s^2}{\Gamma}\left(\left(\frac{\rho}{\rho_0}\right)^{\Gamma} - 1\right), \tag{2.15}$$

where c_s is the artificial sound speed, $\Gamma = 7$, and ρ_0 is the initial reference density. To approximate incompressibility, a large value of artificial speed of sound c_s is chosen, typically resulting in a flow Mach number of $M \approx 0.1$.

2.2.2 Entropically Damped Artificial Compressibility Formulation

Entropically damped artificial compressibility (EDAC) method (Clausen 2013) proposes an alternative way to the weakly-compressible method. The pressure doesn't depend on the density and uses the pressure-velocity system that also employ pressure damping. The pressure evolution equation is thus given by,

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\rho c_s^2 \mathrm{div}(\mathbf{u}) + \nu_{\mathrm{e}}\nabla^2 p, \tag{2.16}$$

where ν_e is the EDAC viscosity parameter.

2.3 SPH Discretization of the Governing Equations

We apply SPH discretization to the Navier-Stokes equations discussed above. In all the simulations we do not keep the density constant, density is computed by the summation density,

$$\rho_i = \sum_j m_j W_{ij} \tag{2.17}$$

where the subscript i denotes the i^{th} particle, $W_{ij} = W(|\mathbf{r}_{ij}|, h_{ij})$ is the SPH smoothing kernel, where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $h_{ij} = (h_i + h_j)/2$, $\mathbf{r}$ is the position of the particle and h the smoothing length of the kernel. The summation is carried over neighbors of the i^{th} particle. The SPH discrete form of the Navier-Stokes using SPH approximation can be written as,

$$\frac{d\rho_i}{dt} = \sum_j m_j (\mathbf{u}_i - \mathbf{u}_j) \cdot \nabla_i W_{ij}, \tag{2.18}$$

$$\frac{d\mathbf{u}_i}{dt} = \mathbf{f}_{\text{p}} + \mathbf{f}_{\text{visc}} + \mathbf{f}_{\text{body}}, \tag{2.19}$$

where ρ_i, $\mathbf{u}_i$ is respectively the density and velocity of the i^{th} particle. $\mathbf{f}_{\text{p}}$ is the acceleration due to pressure gradient commonly discretized (Monaghan 1992) as,

$$\mathbf{f}_{\text{p}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij}, \tag{2.20}$$

where p_i is pressure of i^{th} particle, p_j is pressure, m_j is the mass, and ρ_j is the density of the j^{th} particle, $\nabla_i W_{ij} = \frac{\partial W}{\partial \mathbf{r}_i}(\mathbf{r}_i - \mathbf{r}_j, h)$ is the gradient of the smoothing kernel. For alternative forms of the discretization and an application to astrophysics see Rosswog (2009). $\mathbf{f}_{\text{visc}}$ is the acceleration due to viscous forces (Cleary et al. 1999) and in two dimensions is given by,

$$\mathbf{f}_{\text{visc}} = \sum_j m_j \frac{4\nu}{(\rho_i + \rho_j)} \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{(|\mathbf{r}_{ij}|^2 + \eta h^2)} \mathbf{u}_{ij}, \tag{2.21}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, and $\mathbf{f}_{\text{body}}$ is the acceleration due to external body forces. Finally, the particle positions are obtained by integrating in time the following ODE,

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{u}_i. \tag{2.22}$$

2.4 Time Integration

The equations of motion and momentum equations are then integrated in time to obtain the positions and velocities respectively, a Predictor-Corrector integrator (Monaghan 2005), also known as kick-drift-kick integrator, is used to integrate in time which is described as,

$$\mathbf{u}_i^{n+\frac{1}{2}} = \mathbf{u}_i^n + \frac{\Delta t}{2}\left(\frac{d\mathbf{u}_i}{dt}\right)^n, \tag{2.23}$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t \mathbf{u}_i^{n+\frac{1}{2}}, \tag{2.24}$$

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^{n+\frac{1}{2}} + \frac{\Delta t}{2}\left(\frac{d\mathbf{u}_i}{dt}\right)^{n+1}. \tag{2.25}$$

The PEC integrator is a popular choice in SPH simulations due its symplectic nature that can lead to energy preservation. The use of PEC integrator over Runge-Kutta second order integrator (RK2) can be motivated by the fact that for the PEC integrator only a single computation of acceleration is needed per time-step, whereas RK2 requires two. One can also choose between other available integrators like first order Euler integrator, second-order Runge-Kutta integrators or any other. Often, one looks at the accuracy and efficiency of these integrators. Multi-step integrators like Adams-Bashforth are also available if memory storage is not a concern. Energy conserving integrators like symplectic integrators (Leimkuhler et al. 1996) may also be used. A good list of literature on variational integrators is given in (Marsden et al. 2001). The time-step for the integration is based on a CFL criterion that depends on the differential equations being solved.

2.5 Particle Shifting

Due to the Lagrangian nature of SPH the particle tends to follow the streamlines in the flow. This may not be a desired behaviour for numerical simulation where the evaluation of derivatives require uniform particle distribution. To maintain uniform particle distribution we employ two different types of shifting in our work: position-based shifting, and transport velocity-based shifting.

2.5.1 Position-based Shifting

Position-based shifting was proposed by Xu et al. (2009) to improve stability of the incompressible SPH method for high Reynolds number flows. We use a limited form of the particle shifting technique of Lind et al. (2012) which is based on evaluating the gradient of the kernel function. A particle with an index i at a current position $\mathbf{r}_i$ is shifted to a new position $\mathbf{r}_i'$ as,

$$\mathbf{r}_i' = \mathbf{r}_i + \delta\mathbf{r}_i, \tag{2.26}$$

where,

$$\delta\mathbf{r}_i = -\frac{h_i^2}{2}\sum_j \frac{m_j}{\rho_j}\left(1 + 0.24\left(\frac{W(r_{ij}, h_{ij})}{W(\xi\Delta x, h_{ij})}\right)^4\right)\nabla W_{ij}, \tag{2.27}$$

where ξ is the point of inflection of the kernel (Crespo 2008), and $h_{ij} = (h_i + h_j)/2$. For quintic spline the point of inflection is $\xi = 0.759298480738450$. After shifting we correct the fluid properties by using a Taylor series approximation. Consider a fluid property $\varphi_i = \varphi(\mathbf{r}_i)$ the corrected value $\varphi_i' = \varphi(\mathbf{r}_i')$ is obtained by,

$$\varphi_i' = \varphi_i + (\nabla\varphi)_i \cdot \delta\mathbf{r}_i. \tag{2.28}$$

We use position-based shifting in the DTSPH scheme (chapter 4) and in the adaptive algorithm discussed in chapter 5.

2.5.2 Transport-Velocity-based Shifting

An alternative to the position-based shifting is the transport-velocity-based[1] formulation proposed by Adami et al. (2013). This method falls under the general class of methods called arbitrary Lagrangian-Eulerian framework (Oger et al. 2016b).

The transport velocity formulation introduces another velocity called transport velocity $\tilde{\mathbf{u}}$ with which particles' position are updated,

$$\frac{\tilde{\mathrm{d}}\mathbf{r}}{\mathrm{d}t} = \tilde{\mathbf{u}}. \tag{2.29}$$

where, the transport material derivative on any quantity is given as,

$$\frac{\tilde{\mathrm{d}}f(\mathbf{r}, t)}{\mathrm{d}t} = \frac{\partial f(\mathbf{r}, t)}{\partial t} + \tilde{\mathbf{u}} \cdot \nabla f(\mathbf{r}, t). \tag{2.30}$$

[1] See Sun et al. (2019b) for a discussion on the particle shifting techniques.

The transport velocity introduces additional terms to the Navier-Stokes equations (Adepu et al. 2021). The transport velocity is defined by,

$$\tilde{\mathbf{u}}(\mathbf{r}, t+\delta t) = \mathbf{u}(\mathbf{r}, t) + \delta t \left(\frac{\tilde{\mathrm{d}}\mathbf{u}}{\mathrm{d}t} - p_i^0 \sum_j \frac{m_j}{\rho_j^2} \nabla W(\mathbf{r}_{ij}, h) \right), \tag{2.31}$$

where, the second term in the RHS is the regularization force. This form of regularization force can be changed, for instance,

$$\tilde{\mathbf{u}}(\mathbf{r}, t+\delta t) = \mathbf{u}(\mathbf{r}, t) + \left(\delta t \frac{\tilde{\mathrm{d}}\mathbf{u}}{\mathrm{d}t} + \frac{\delta \mathbf{r}_i}{\delta t} \right). \tag{2.32}$$

where, $\delta \mathbf{r}_i$ is from eq. (2.27). We use the generalized transport velocity formulation proposed by Zhang et al. (2017) in the next chapter, and in the EDAC formulation in chapter 5.

2.6 Boundary Conditions

We employ periodic, no-slip, free-slip, no-penetration and the inlet-outlet boundary conditions in our test cases. We enforce periodic boundary conditions by the use of ghost particles onto which the properties are directly copied from the particles exiting the domain through a periodic boundary.

For the no-slip, free-slip and no-penetration boundary conditions we use the dummy particle technique of Adami et al. (2012). Dummy particles placed in uniform layers are used to discretize the wall. The no-penetration is implicitly enforced by using the wall velocity in the EDAC equation (Adami et al. 2012). For the no-slip or free-slip we extrapolate the values of velocity of the fluid onto the dummy wall particles by,

$$\mathbf{u}_w = 2\mathbf{u}_i - \hat{\mathbf{u}}_i, \tag{2.33}$$

where the subscript w denotes the dummy wall particles, $\mathbf{u}_i$ is the prescribed wall velocity, and

$$\hat{\mathbf{u}}_i = \frac{\sum_j \mathbf{u}_j W(r_{ij}, h_{ij})}{\sum_j W(r_{ij}, h_{ij})} \tag{2.34}$$

is the Shepard extrapolated velocity of the fluid particles indexed by j onto the dummy wall particles i. The pressure on the wall is calculated from the fluid, to accurately impose the pressure gradient, by,

$$p_w = \frac{\sum_f p_f W(r_{wf}, h_{wf}) + (\mathbf{g} - \mathbf{f}_w) \cdot \sum_f \rho_f \mathbf{r}_{wf} W(r_{wf}, h_{wf})}{\sum_f W(r_{wf}, h_{wf})}, \tag{2.35}$$

where the subscript f denotes the fluid particles, $\mathbf{f}_w$ is the acceleration of the wall, $r_{wf} = |\mathbf{r}_w - \mathbf{r}_f|$, and $h_{wf} = (h_w + h_f)/2$. In the case of free-surface flows, we ensure that the wall does not have any negative pressures as this often causes particles to stick to and sometimes penetrate the body. Hence if the pressure on the wall is negative, we set it to zero. For the inlet-outlet boundary conditions we use the "do-nothing" boundary condition shown in Negi et al. (2019) and the non-reflecting boundary conditions of Lastiwka et al. (2009), the are further elaborated in the chapters.

2.7 PySPH: A Python-based SPH Framework

PySPH is an open-source, Python-based, multi-CPU, and GPU framework for SPH simulations. We have used the interface PySPH provides to implement most of our adaptive algorithm. PySPH exposes a Python-friendly interface to the user and using internal mechanism generates a high performance serial or parallel code based on the choice of the user. These functionalities enable one to write the code without having to pay much attention to the low-level high-performance complexities of implementing on different platforms like multi-core CPUs, or GPUs. However, it is useful to note that PySPH does support iterative solution of matrices using a matrix-free formulation. For an overview of the PySPH design, see Ramachandran et al. (2021c) and `https://pysph.readthedocs.io` for a detailed reference.

3 Simple Iterative Incompressible Smoothed Particle Hydrodynamics

Uh-oh — I've let the cat out of the bag. Let me, then, straightforwardly state the thesis I shall now elaborate: Making variations on a theme is really the crux of creativity.

— Douglas R. Hofstadter, *Metamagical Themas*

3.1 Introduction

In this chapter, we consider the projection-based ISPH scheme of Cummins et al. (1999). We use an iterative approach to solve the PPE as contrasted with the EISPH approach. This makes the formulation completely matrix-free, and easy to derive and implement. This approach is not new and is similar to the approach used in the EISPH (Barcarolo 2013; Basser et al. 2017; Hosseini et al. 2007; Nomeritae et al. 2016) with the key difference being that we iteratively solve the same PPE. This makes a significant difference when we simulate problems at higher Reynolds numbers. We use a weighted Jacobi method to accelerate the convergence of our iterations. Our implementation is efficient, and works comfortably on a GPU architecture. This reformulation also allows us to implement boundary relatively easily as is done in WCSPH schemes and this is harder to do with the traditional ISPH. We note that despite the existence of the similar EISPH for many years, the traditional approach of solving the system of equations using a separate linear solver continues to be popular (Chow et al. 2018). In this work we apply the GTVF to

work with ISPH schemes. Since the time-steps are larger, we use a small modification to move the particles during the regularization.

In order to handle solid boundaries accurately, we modify the method proposed by Adami et al. (2012) to work with the ISPH schemes. The original method does not work too well with the ISPH due to the larger time-steps that the method allows. We provide a simple way to compute the normals on a solid body with just the point information, and use this to improve the solid boundary condition. The resulting method works effectively for a variety of test cases.

Our entire implementation is open source. We employ the open source PySPH (Ramachandran et al. 2021b) framework for the implementation of the scheme. In the interest of reproducible research, the entire work of this chapter is made to be reproducible. Every figure presented in the results section of this chapter is automated (Ramachandran 2018) and the code for the computations is made available at https://gitlab.com/pypr/sisph.

3.2 Incompressible Fluid Dynamics

The basic formulation is that of Cummins et al. (1999). For an incompressible fluid, the continuity equation is,

$$\nabla \cdot \mathbf{u} = 0 \tag{3.1}$$

where $\mathbf{u}$ is the velocity field, and the momentum equation is given by,

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot (\nabla \mathbf{u}) + \mathbf{f} \tag{3.2}$$

where $\frac{d\mathbf{u}}{dt}$ is the material derivative of the velocity field, ρ is density, p is pressure, ν is kinematic viscosity and $\mathbf{f}$ is the external force.

Pressure is obtained by the projection method, which uses Hodge decomposition to project any velocity field, $\mathbf{u}^*$ into a divergence-free component, and a curl-free component,

$$\mathbf{u}^* = \mathbf{u} + \frac{\Delta t}{\rho}\nabla p \tag{3.3}$$

where $\mathbf{u}$ is the divergence free velocity field, ∇p is the curl-free component. The pressure is then obtained by taking divergence of eq. (3.3), which reads,

$$\nabla \cdot (\frac{1}{\rho}\nabla p) = \frac{\nabla \cdot \mathbf{u}^*}{\Delta t} \tag{3.4}$$

where $\mathbf{u}^*$ is the intermediate velocity obtained by integrating the momentum equation without considering the pressure gradient terms (Cummins et al. 1999). After solving for the pressure, p (curl-free component), the divergence free velocity $\mathbf{u}$ is obtained by subtracting the gradient of pressure from the intermediate velocity $\mathbf{u}^*$.

3.3 Incompressible SPH Method

The ISPH like most SPH methods suffers from particle disorder. We consider the Generalized Transport Velocity Formulation (GTVF) by Zhang et al. (2017) to remedy the particle disorder. GTVF extends the Transport Velocity Formulation (TVF) by Adami et al. (2013) to free surface flows and solid dynamics by using variable background pressure instead of a constant global background pressure. In the GTVF formulation, positions and velocity are advected using the transport velocity, which introduces an additional artificial stress term to the momentum equation. The transport velocity is regulated by variable background pressure, which keeps the particles in a uniform configuration. The momentum equation for GTVF is,

$$\frac{\tilde{d}\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla p + \nu\nabla\cdot(\nabla\mathbf{u}) + \frac{1}{\rho}\nabla\cdot\mathbf{A} + \mathbf{f} \tag{3.5}$$

where $\frac{\tilde{d}\mathbf{u}}{dt} = \frac{\partial\mathbf{u}}{\partial t} + \tilde{\mathbf{u}}\cdot\nabla\mathbf{u}$ is the material derivative of velocity field which is advected by the transport velocity, $\tilde{\mathbf{u}}$, and $\mathbf{A} = \rho\mathbf{u}\otimes(\tilde{\mathbf{u}} - \mathbf{u})$ is the artificial stress term.

The SPH discretization of the momentum eq. (3.5) is given by,

$$\frac{\tilde{d}\mathbf{u}_i}{dt} = \mathbf{f}_\mathrm{p} + \mathbf{f}_\mathrm{visc} + \mathbf{f}_\mathrm{b} + \mathbf{f}_\mathrm{avisc} + \mathbf{f}_\mathrm{astress} \tag{3.6}$$

where $\mathbf{f}_\mathrm{p}$ is the force due to pressure gradient, $\mathbf{f}_\mathrm{visc}$ is the force due to viscous forces, $\mathbf{f}_\mathrm{b}$ is the body force, $\mathbf{f}_\mathrm{avisc}$ is the force due to artificial viscosity and $\mathbf{f}_\mathrm{astress}$ is the force due to artificial stress which is a consequence of the GTVF formulation.

In the literature there are two ways of computing pressure gradient. One is a symmetric, momentum-preserving form (Cummins et al. 1999; Hu et al. 2007) and the other is asymmetric and does not preserve momentum (Lind et al. 2012; Skillen et al. 2013; Xu et al. 2009). In the present work the symmetric form is labeled as “symm” and the asymmetric form as “asymm”. We have used both the forms in this work and report the differences between them in our results.

The SPH discretization of the symmetric form (Monaghan 1992) is,

$$\mathbf{f}_{\text{p, symm}} = -\sum_{j \in N(i)} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \tag{3.7}$$

and the asymmetric form (Monaghan 1992) is,

$$\mathbf{f}_{\text{p, asymm}} = -\sum_{j \in N(i)} \frac{m_j}{\rho_i \rho_j} (p_j - p_i) \nabla W_{ij} \tag{3.8}$$

where ∇W_{ij} is the gradient of the smoothing kernel. Viscous forces are discretized as (Cummins et al. 1999),

$$\mathbf{f}_{\text{visc}} = \sum_{j \in N(i)} m_j \frac{4\nu}{(\rho_i + \rho_j)} \frac{\mathbf{r}_{ij} \cdot \nabla W_{ij}}{(|\mathbf{r}_{ij}|^2 + \eta h_{ij}^2)} \mathbf{u}_{ij} \tag{3.9}$$

where $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$, ν is the kinematic viscosity, $\eta = 0.01$. Artificial viscosity (Monaghan 2005) is added to the momentum equation wherever necessary given by,

$$\mathbf{f}_{\text{avisc}} = \sum_{j \in N(i)} \Pi_{ij} \nabla W_{ij} \tag{3.10}$$

where, Π_{ij} is computed by,

$$\Pi_{ij} = \begin{cases} \frac{-\alpha h_{ij} \bar{c}_{ij} \phi_{ij}}{\bar{\rho}_{ij}}, & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ 0, & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} \geq 0. \end{cases} \tag{3.11}$$

The artificial stress due to the GTVF formulation (Zhang et al. 2017) is evaluated using,

$$\mathbf{f}_{\text{astress}} = \sum_{j \in N(i)} \left(\frac{\mathbf{A}_i}{\rho_i^2} + \frac{\mathbf{A}_j}{\rho_j^2} \right) \tag{3.12}$$

where, $A_i = \rho_i \mathbf{u}_i \otimes (\tilde{\mathbf{u}}_i - \mathbf{u}_i)$. The transport velocity, $\tilde{\mathbf{u}}$ due to GTVF (Zhang et al. 2017) is updated by adding an additional background pressure to the simulation,

$$\tilde{\mathbf{u}}_i^{n+1} = \mathbf{u}_i + \Delta t \left(\frac{\tilde{d}\mathbf{u}_i}{dt} + \mathbf{f}_{i,\text{GTVF}} \right) \tag{3.13}$$

where, $\frac{\tilde{d}\mathbf{u}_i}{dt}$ is obtained from eq. (3.6) and

$$\mathbf{f}_{i,\text{GTVF}} = -p_i^0 \sum_{j \in N(i)} \frac{m_j}{\rho_j^2} \nabla W(\mathbf{r}_{ij}, \tilde{h}_{ij}). \tag{3.14}$$

On numerical investigation, we suggest $\tilde{h} = 0.5h_{ij}$ for external and free surface flows, and $\tilde{h}_{ij} = h_{ij}$ for internal flows, $p_i^0 = \min(10|p_i|, p_{\text{ref}})$ for external flows, and $p_i^0 = p_{\text{ref}}$ is kept constant for internal flows. The choice of reference pressure, p_{ref}, is typically taken to be ρc^2 for external flows and $2\max(p_i)$, i.e., twice the maximum pressure in the entire flow after the first iteration for internal flows. Here, we assume that $c = 10|\mathbf{u}|_{\max}$ and this is simply a reference pressure.

The pressure-Poisson equation that is solved is,

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p\right)_i = \frac{\nabla \cdot \mathbf{u}^*{}_i}{\Delta t} \tag{3.15}$$

where Δt is the time-step. The approximate projection form for the PPE operator in eq. (3.15) given by Cummins et al. (1999) is used, resulting in the discretized form as,

$$\sum_j \left(\frac{4m_j}{\rho_i(\rho_i+\rho_j)}\right)\frac{p_{ij}\mathbf{r}_{ij}\cdot\nabla_i W_{ij}}{|r_{ij}^2|+\eta h_{ij}^2} = \sum_j -\frac{m_j}{\rho_j \Delta t}\mathbf{u}^*{}_{ij}\cdot\nabla_i W_{ij}, \tag{3.16}$$

where $p_{ij} = p_i - p_j$. In the next section we describe the matrix-free setup of the PPE.

3.3.1 Time Integration

Due to its simplicity we use the time integration as given by Cummins et al. (1999). However, the higher order method proposed in Nair et al. (2015) could also be used. Time integration is performed by beginning with calculating the intermediate position using the transport velocity at the current time,

$$\mathbf{r}_i^* = \mathbf{r}_i^n + \Delta t\tilde{\mathbf{u}}_i^n, \tag{3.17}$$

then the intermediate velocity is obtained by integrating the momentum equation neglecting the force due to the pressure gradient,

$$\mathbf{u}_i^* = \mathbf{u}_i^n + \Delta t(\mathbf{f}_{i,\text{visc}}^n + \mathbf{f}_{i,\text{avisc}}^n + \mathbf{f}_{i,\text{astress}}^n + \mathbf{f}_{i,\text{body}}^n). \tag{3.18}$$

After that, the pressure-Poisson equation given by eq. (3.16) is solved to obtain the pressure. Lastly, the divergence free velocity is then found by correcting the intermediate velocity as,

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^* + \Delta t\mathbf{f}_{i,\text{p}}^n. \tag{3.19}$$

We use modified GTVF to redistribute particles. The GTVF formulation is first used in the context of weakly-compressible flows, where the time step is much lower than ISPH schemes.

The time-steps are large in the ISPH and applying the GTVF correction in one time-step introduces stability issues. We modify the GTVF formulation by applying the GTVF force ($\mathbf{f}_{\text{GTVF}}$, eq. (3.14)) to shift the particles iteratively in one time step. We then compute the transport velocity using the initial and final positions after the iteration.

The GTVF acceleration is a function of positions $\mathbf{r}$, background pressure p_{ref} and smoothing length h_{ij}. In all our simulations h_{ij} is fixed and for a given time the background pressure remains unchanged. We use K sub-time-steps to integrate the GTVF accelerations without recomputing any neighbors such that $\Delta\tau = \Delta t/K$. If we denote $\mathbf{r}^k, \tilde{\mathbf{u}}^k$ as the position and GTVF velocity of a particle at the k'th sub-iterate, we first start with $\mathbf{r}^0$ set to the current particle position, and set $\tilde{\mathbf{u}}^0 = 0$.

$$\mathbf{r}^k = \mathbf{r}^{(k-1)} + \Delta\tau\tilde{\mathbf{u}}^{k-1} + \frac{1}{2}(\Delta\tau)^2\mathbf{f}_{\text{GTVF}}(\mathbf{r}^{k-1}), \tag{3.20}$$

$$\tilde{\mathbf{u}}^k = \tilde{\mathbf{u}}^{(k-1)} + \Delta\tau\mathbf{f}_{\text{GTVF}}(\mathbf{r}^{(k-1)}). \tag{3.21}$$

After the GTVF iterative step is done, we compute the transport velocity by looking at the initial and final positions

$$\tilde{\mathbf{u}}_i^{n+1} = \mathbf{u}_i^{n+1} + \frac{\mathbf{r}^K - \mathbf{r}^0}{\Delta t}, \tag{3.22}$$

particles are then advected to the next time step using transport velocity of current and previous time steps given by,

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t\left(\frac{\tilde{\mathbf{u}}_i^{n+1} + \tilde{\mathbf{u}}_i^n}{2}\right). \tag{3.23}$$

3.4 Iterative Formulation

Traditionally, eq. (3.16) is solved by setting up a sparse matrix corresponding to the coefficients of p_i and p_j in the equation and computing the right-hand side. This system is then solved using any fast, sparse matrix solver. In the EISPH (Barcarolo 2013; Basser et al. 2017; Hosseini et al. 2007; Nomeritae et al. 2016) a simple explicit formula is provided to update the pressure.

We use this approach to write this system in terms of a Jacobi iteration we introduce a time index, k, and write p^k as the pressure at the k^{th} iteration. We rewrite eq. (3.16) in the form,

$$D_{ii}p_i^{k+1} = \text{RHS}_i - \sum_j \text{OD}_{ij}p_j^k, \tag{3.24}$$

where OD_{ij} represents the off-diagonal coefficients which are multiplied by pressure at the earlier iteration p^k and RHS_i is the right-hand-side of eq. (3.16). We can easily write the following expressions for the coefficients in eq. (3.24) as,

$$D_{ii} = \sum_j \left(\frac{4m_j}{\rho_i(\rho_i + \rho_j)} \right) \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{|r_{ij}^2| + \eta^2} \tag{3.25}$$

$$\text{OD}_{ij} = \left(\frac{-4m_j}{\rho_i(\rho_i + \rho_j)} \right) \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{|r_{ij}^2| + \eta^2} \tag{3.26}$$

To begin with, we set $p^{k=0}(t + \Delta t) = p(t)$, that is our first iteration starts with the existing value of pressure of the particle. We find the pressure for the next iteration using weighted Jacobi method[1] as,

$$p_i^{k+1} = \omega \frac{(\text{RHS}_i - \sum_j \text{OD}_{ij} p_j^k)}{D_{ii}} + (1 - \omega) p_i^k \tag{3.27}$$

The default value of $\omega = 0.5$. We note that the entire term $\sum_j \text{OD}_{ij} p_j^k$, needs to be accumulated into a single term and hence only requires a single number per particle for storage. Thus, per particle we require storing three terms, viz. D_{ii}, $\sum_j \text{OD}_{ij} p_j^k$, and RHS_i. We note that if a particle has no neighbors, then D_{ii} can be zero, in which case we set the pressure of the particle to zero. Similarly, to handle free surfaces we compute the summation density of the particles and if the ratio $\rho_i/\rho_0 < 0.8$, where ρ_0 is the reference or rest density, we treat the particle as a free-surface particle and set its pressure to zero. Note that during these pressure iterations we do not move the particles. We continue to iterate until the relative change in the pressure is less than some user-specified amount,

$$\sum_i \frac{|p_i^{k+1} - p_i^k|}{\max(\Lambda,\ \sum_i |p_i^{k+1}|)} < \epsilon, \tag{3.28}$$

where $\Lambda = \max(RHS_i/D_{ii})$. When the mean pressure in the flow is less than one, we use the largest value of RHS_i/D_{ii} as a measure of the mean pressure and this conveniently avoids zeros in the denominator while also using a relative scale for the error. The minimum number of iterations performed is set to two.

We vary the tolerance parameter, ϵ, in our numerical studies. We consider the suitable choice of this tolerance parameter next.

[1] In the original paper (Muta et al. 2020b), of which this thesis is partly-based, weighted Jacobi method is incorrectly written as Successive-Over-Relaxation (SOR).

3.4.1 Choice of the Tolerance Parameter

While iterating for convergence, we note that using very small values of ϵ is not necessary. We note that the discretized terms in eq. (3.25) involve errors of at best $O(h^2)$ so there is no major advantage to solving the linear system to high accuracy.

Furthermore, for many time-steps, the pressure difference at a point between two time-steps is a small quantity and since we start with an accurate pressure, we expect that obtaining the new pressure will take very few iterations. By setting a tolerance of say 0.001, we ensure that if there are any changes to the pressure during the Jacobi iterations that are less than 0.1% that we stop iterations. In practice, we find that most often only a few iterations (typically 2 to 10) are necessary. It is important to note that Chorin (Chorin 1968), in his original fractional step work, makes a similar suggestion. In addition as mentioned in the introduction, the EISPH (Barcarolo 2013; Basser et al. 2017; Hosseini et al. 2007; Nomeritae et al. 2016) only uses a single iteration and obtains acceptable results for a wide variety of problems. However, in the present work, we find that a single iteration is insufficient especially in the case of high-Reynolds number problems and we explore this in Section 3.7.2. As such, we recommend choosing the ϵ to be of $O(h)$ or $O(\Delta t)$.

Table 3.1, we show the average number of Jacobi iterations for different problems and tolerances. These results seem to show that using Jacobi iterations is very effective. Although we do

Table 3.1: Average number of iteration taken for Jacobi for various tolerances and problems.

Problem	Tolerance	Jacobi (avg no of. iterations)
Cavity	0.001	3.0
Dam-break 2D	0.001	5.0
Cavity	0.01	2.0
Dam-break 2D	0.01	2.0

not discuss this in detail, our repository contains an implementation of a BiCGStab (Vorst 1992) iterative algorithm to solve the PPE. A BiCGStab is needed since when we apply the boundary conditions, the resulting matrix may become non-symmetric. We find that implementing the BiCGStab correctly is much more complex as compared to the Jacobi iterations. Further, even though it converges faster, we find that the iterations do not seem to produce significantly better results. We find that in practice, it is much faster to use the Jacobi approach since most of the

Algorithm 1 Simple Iterative ISPH algorithm

while $t < t_{\text{final}}$ **do**
  Compute ρ_i using eq. (2.17).
  Predict position $\mathbf{r}_i^* \leftarrow \mathbf{r}_i^n + \Delta t \tilde{\mathbf{u}}^i$
  Compute $\mathbf{f}_{i,\text{visc}}$, $\mathbf{f}_{i,\text{avisc}}$, $\mathbf{f}_{i,\text{b}}$, and $\mathbf{f}_{i,\text{astress}}$.
  Predict velocity $\mathbf{u}_i^* \leftarrow \mathbf{u}_i^n + \Delta t(\mathbf{f}_{i,\text{visc}} + \mathbf{f}_{i,\text{b}} + \mathbf{f}_{i,\text{avisc}} + \mathbf{f}_{i,\text{astress}})$
  Compute $\nabla \cdot \mathbf{u}_i^*$ from RHS of eq. (3.16).
  while $\sum_i \frac{|p_i^{k+1} - p_i^k|}{\max(1, \sum_i |p_i^{k+1}|)} < \epsilon$ **do**
    Compute D_{ii} and OD_{ij} .
    Compute p_i^{k+1} using eq. (3.27).
  Compute $\mathbf{f}_{i,\text{p}}$ either from equation eq. (3.8) or eq. (3.7).
  Correct velocity $\mathbf{u}_i^{n+1} \leftarrow \mathbf{u}_i^* + \Delta t\ \mathbf{f}_{i,\text{p}}$.
  Set $k \leftarrow 0$, $\mathbf{r}^{k=0} \leftarrow \mathbf{r}^*$, $\tilde{\mathbf{u}}^0 \leftarrow 0$
  while $k < K$ **do**
    Compute $\mathbf{f}_{i,\text{GTVF}}(\mathbf{r}^k)$ using eq. (3.14).
    Compute $\mathbf{r}^{k+1}$ using eq. (3.20)
    Compute $\tilde{\mathbf{u}}^{k+1}$ using eq. (3.21)
  Update GTVF velocity to $\tilde{\mathbf{u}}_i^{n+1} \leftarrow \mathbf{u}_i^{n+1} + \frac{\mathbf{r}^K - \mathbf{r}^0}{\Delta t}$.
  Update the positions $\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + 0.5\Delta t(\tilde{\mathbf{u}}_i^{n+1} + \tilde{\mathbf{u}}_i^n)$

time we only need a few iterations to converge. This suggests that using more accurate iterative sparse linear solvers may be unnecessary.

We therefore recommend the use of the Jacobi iterations approach for its efficiency and simplicity.

3.4.2 The Algorithm

For clarity, the (algorithm 1) shows the steps involved in the proposed scheme. This simple formulation works very well in practice as borne out by our numerous benchmarks. The formulation allows us to satisfy boundary conditions that are traditionally not easy to do with a matrix-based formulation.

The time steps are chosen based on the following criteria, the CFL criterion is,

$$\Delta t_{\text{cfl}} = 0.25 \frac{h_{ij}}{|\mathbf{U}|} \tag{3.29}$$

where $|\mathbf{U}|$ is the magnitude of maximum velocity in the simulation, the viscous criterion is,

$$\Delta t_{\text{visc}} = 0.25 \frac{h_{ij}^2}{\nu} \tag{3.30}$$

the condition due to external force (if applicable) is,

$$\Delta t_{\text{force}} = 0.25 \sqrt{\frac{h_{ij}}{g}} \tag{3.31}$$

time steps are then chosen to be the minimum of above conditions,

$$\Delta t = \min \left(\Delta t_{\text{cfl}}, \Delta t_{\text{visc}}, \Delta t_{\text{force}} \right) \tag{3.32}$$

Adaptive time steps can be used in the following way. At each time step the maximum velocity in the simulation is calculated, and used in the CFL condition,

$$\Delta t_{\text{cfl}}^{n+1} = 0.25 \frac{h_{ij}}{\max |\mathbf{u}_i^n|} \tag{3.33}$$

where $\max |\mathbf{u}_i^n|$ is the maximum magnitude of velocity in the domain at the current time step. Similarly, the time step restriction due to force is calculated as,

$$\Delta t_{\text{force}}^{n+1} = 0.25 \sqrt{\frac{h_{ij}}{\max |\mathbf{f}_i^n|}} \tag{3.34}$$

where $\mathbf{f}_i^n = \mathbf{f}_{\text{p}}^n + \mathbf{f}_{\text{visc}}^n + \mathbf{f}_{\text{body}}^n + \mathbf{f}_{\text{avisc}}^n + \mathbf{f}_{\text{astress}}^n$. There is no change in $\Delta t_{\text{visc}}^{n+1}$ as viscosity is held constant in all our simulations, hence the time step for the next time iteration is,

$$\Delta t^{n+1} = \min \left(\Delta t_{\text{cfl}}^{n+1}, \Delta t_{\text{visc}}^{n+1}, \Delta t_{\text{force}}^{n+1} \right) \tag{3.35}$$

3.5 Solid Wall Boundary Conditions

In order to satisfy solid wall boundary conditions, the method proposed by Adami et al. (2012) is slightly modified to work with the ISPH scheme. This method uses 3 to 4 layers of dummy

(or ghost) solid particles and then performs a Shepard interpolation of the pressure and also accounts for any fluid acceleration to set the solid wall pressure. As discussed in the original paper (Adami et al. 2012) the pressure of the dummy wall particles are set using,

$$p_w = \frac{\sum_f p_f W_{wf} + (\mathbf{g} - \mathbf{a}_w) \cdot \sum_f \rho_f \mathbf{r}_{wf} W_{wf}}{\sum_f W_{wf}}, \tag{3.36}$$

where the subscript w denotes a wall particle and f a fluid particle, $\mathbf{a}_w$ denotes the acceleration of the wall and $\mathbf{g}$ the acceleration due to gravity if relevant. In the case of free-surface flows, we ensure that the wall does not have any negative pressures as this often causes particles to stick to and sometimes penetrate the body. Hence if the pressure on the wall is negative, we set it to zero.

In our iterative scheme to solve for the pressure, during each iteration, we perform a Shepard interpolation of the fluid pressure using eq. (3.36) and use those values in evaluating the diagonal and off-diagonal terms. Thus the solid particles are treated just as fluids but with their pressure evaluated based on the extrapolated fluid pressure from the previous iteration. The dummy particle velocities are normally set by calculating a Shepard-extrapolated fluid velocity on the ghost particles using,

$$\tilde{\mathbf{u}}_i = \frac{\sum_j \mathbf{u}_j W_{ij}(h_{ij}/2)}{\sum_j W_{ij}(h_{ij}/2)}, \tag{3.37}$$

where $h_{ij} = (h_i + h_j)/2$. Note that we use a smaller kernel radius than the original scheme to ensure that the closest fluid particles have the most influence on the wall velocity. The velocity of the dummy particles is then set by,

$$\mathbf{u}_w = 2\mathbf{u}_i - \tilde{\mathbf{u}}_i, \tag{3.38}$$

where $\mathbf{u}_i$ is the physical velocity of the wall itself. If a no-slip boundary condition is satisfied, the above wall velocity is used for the wall particles when calculating the viscous forces. For the ISPH method, due to the larger allowed time-steps, this basic approach generally leads to some amount of leakage. We modify this boundary condition and ensure that the resulting ghost velocity $\mathbf{u}_w$ has no component into the solid itself. This is done by using the normal of the solids. We compute the normal of the solid bodies and if the component of the ghost velocity in the direction of the normal is into the solid, we subtract that component,

$$\mathbf{u}_w = \mathbf{u}_w - (\mathbf{u}_w \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} \tag{3.39}$$

Note that this is not done when the direction of the dummy particle velocity is away from the solid body.

When solving the PPE, the boundary condition on $\mathbf{u}^*$ is typically set such that the walls do not have any slip. However, we find that when simulating high-Reynolds number flows this introduces unnecessary divergence on the fluid particles leading to noisy results. This can be mitigated by using a slip velocity for the $\mathbf{u}^*$. This is illustrated for the case of the lid-driven cavity problem in Section 3.7.3.

3.5.1 Inlet and Outlet Boundary Conditions

It is relatively straightforward to simulate wind-tunnel type problems with our scheme as well. This requires that the inlet and outlet boundary conditions be set appropriately.

For the inlet, we set the prescribed velocity, this is often a constant or a particular velocity profile. The pressure of the inlet region is also solved for, based on the iterative pressure-Poisson equation.

For the outlets we use a modified "do-nothing" type condition as elaborated in Negi et al. (2019). As a fluid particle enters the outlet region, its properties $\mathbf{u}$ and p are frozen. The particle is advected in the outlet region using a Shepard extrapolated velocity from the current fluid particles along the direction of the outlet. For example if the outlet is oriented perpendicular to the x-axis, then we take the fluid's x-component of velocity $\mathbf{u}$ and perform a Shepard interpolation of that on the fluid at each time-step and move the outlet particles. This simple approach works very well as seen from the results presented later.

3.6 Computation of Normals

The normal is itself computed purely using the particle positions in a novel way by first computing the following quantity for the solid particles,

$$\mathbf{n}_i^* = \sum_j -\frac{m_j}{\rho_j}\nabla_i W_{ij} \tag{3.40}$$

If the magnitude of the resulting vector is less than $\frac{1}{4h_i}$, then the $\mathbf{n}^*$ is set to zero otherwise we normalize the vector by its magnitude. Then, we smooth these normals using an SPH approxi-

mation,

$$\mathbf{n}_i = \sum_j \frac{m_j}{\rho_j} \mathbf{n}_j^* W_{ij} \tag{3.41}$$

These normal vectors are then made into unit normals. This produces smooth normals with the positions of the particle alone.

3.7 Results and Discussion

We simulate several standard benchmarks illustrating the proposed method. These benchmarks include internal flows with exact solutions, external flows with a free-surface, two and three dimensional cases and finally the flow past a circular cylinder. We also demonstrate the performance of the new scheme on a GPU. Where possible we compare our results with those produced using the traditional ISPH implementations, and with weakly-compressible SPH using the TVF (Adami et al. 2013) or EDAC-SPH (Ramachandran et al. 2019) formulations.

3.7.1 Taylor-Green Vortex

The Taylor-Green vortex in two dimensions is periodic and has an exact solution given by,

$$u = -Ue^{bt}\cos(2\pi x)\sin(2\pi y) \tag{3.42}$$

$$v = \quad Ue^{bt}\sin(2\pi x)\cos(2\pi y) \tag{3.43}$$

$$p = -U^2 e^{2bt}(\cos(4\pi x) + \cos(4\pi y))/4, \tag{3.44}$$

where $U = 1m/s$ and $b = -8\pi^2/Re$, $Re = UL/\nu$ and $L = 1m$.

We simulate this problem by setting the initial condition at $t = 0$ for a given Re. We choose $Re = 100$ for our initial tests and compare the results with the exact solution. We use the quintic spline with $h/\Delta x = 1.0$.

The decay rate of the velocity is computed by plotting the maximum velocity $|\mathbf{u}_{\max}|$ at each time. We compute the L_1 error in the velocity magnitude as,

$$L_1 = \frac{\sum_i |\mathbf{u}_{i,computed}| - |\mathbf{u}_{i,exact}|}{\sum_i |\mathbf{u}_{i,exact}|}, \tag{3.45}$$

where $\mathbf{u}_{i,exact}$ is found at the position of the i^{th} particle. We compute the L_1 error in the pressure using,

$$p_{L_1} = \frac{\sum_i |p_{i,computed} - p_{i,avg} - p_{i,exact}|}{\max_i(p_{i,exact})}. \tag{3.46}$$

Here $p_{i,avg}$ is the average pressure due to the particle and its neighbors. This is done to avoid any constant pressure bias in any of the schemes.

Since the Taylor-Green vortex has an exact solution, we also take the opportunity to test various parameters in the new scheme. In particular we explore variations in the following,

- Comparison with the original matrix-based formulation of ISPH, with and without the use of shifting, vs. GTVF and perturbation.
- Different tolerance value, ϵ.
- The use of the symmetric form of the pressure gradient.
- Two different resolutions with different Reynolds numbers.

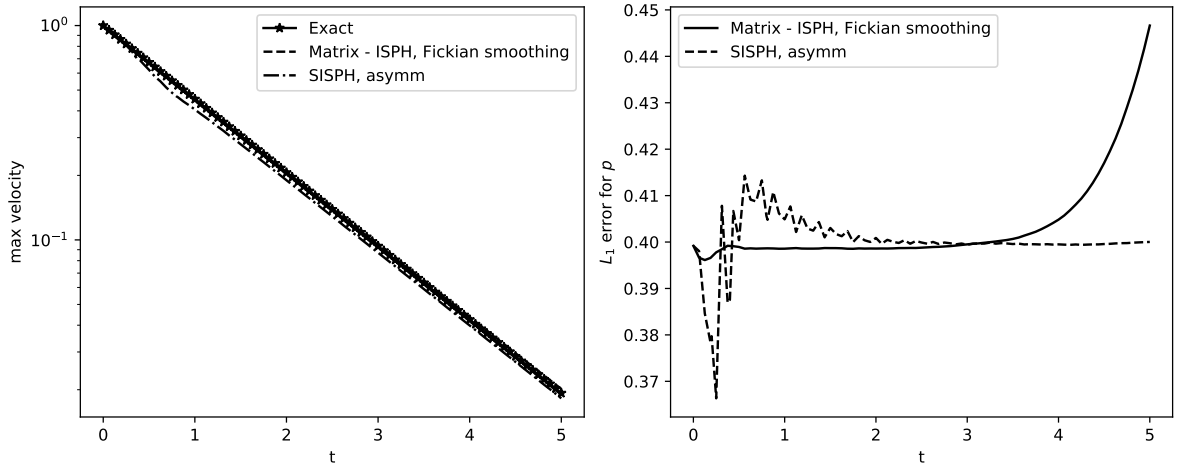
We then compare the results with the WCSPH scheme, the IISPH scheme and the EDAC scheme. We add a small amount of perturbation (of at most $\Delta x/5$), in the initial position of the particles for these schemes to ensure uniform particle distribution. But, the mass and density remain unchanged.

3.7.1.1 Comparison with ISPH

We compare the proposed scheme hereinafter called as SISPH, to the original matrix form of ISPH (Cummins et al. 1999), we use a particle shifting technique (Xu et al. 2009) to maintain uniform particle distribution, as without particle shifting the ISPH method is unstable for higher Reynolds number due to particle disorder (Xu et al. 2009). Here the tolerance is set to, $\epsilon = 10^{-2}$. The "asymm" form of pressure gradient is used. For the pressure coefficient matrix in ISPH we construct a sparse matrix, and use bi-conjugate gradient stabilized method from SciPy (Virtanen et al. 2020) sparse matrix library to solve the pressure-Poisson matrix.

Figure 3.1, the decay of the velocity and the error in the pressure are plotted. As can be seen, the decay in velocity of the new scheme is close to the exact solution. The pressure plots also reveal that the new scheme performs very well. Figure 3.2 shows the particle plots for ISPH, ISPH with Fickian diffusion based shifting (Lind et al. 2012; Skillen et al. 2013), and SISPH

schemes. As said before ISPH is unstable without shifting (Xu et al. 2009). ISPH with Fickian based shifting and SISPH both result in a uniform distribution of particles. This shows that the new scheme performs better than the original ISPH and is comparable with the case of ISPH along with shifting.



(a) Maximum velocity decay vs time.

(b) p_{L_1} vs time.

Figure 3.1: SISPH scheme compared with Matrix based ISPH method with shifting.

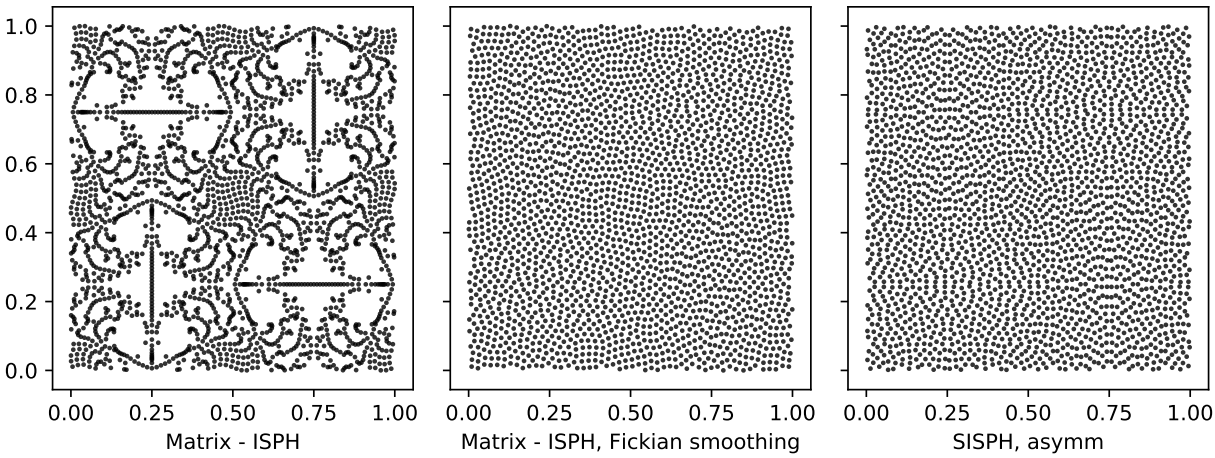
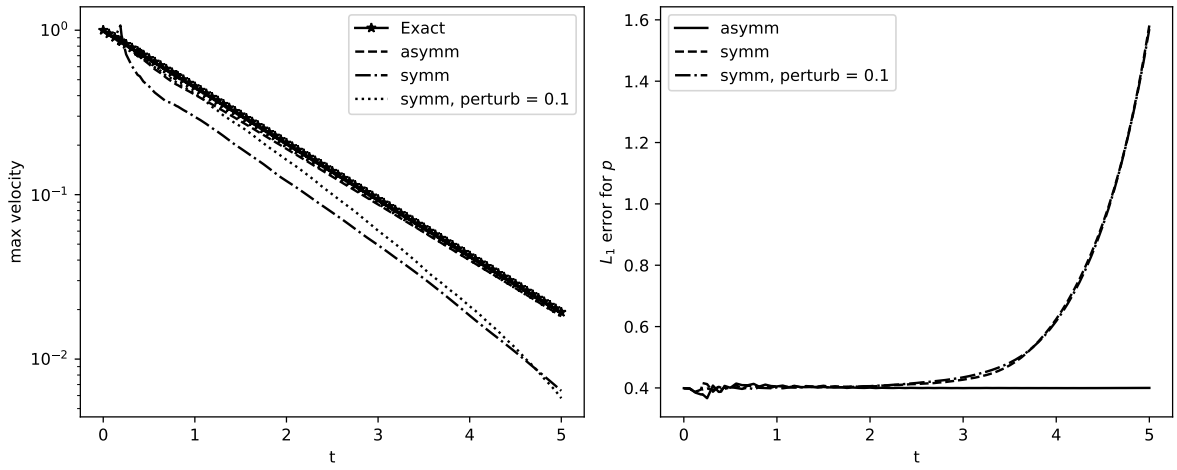


Figure 3.2: Particle plots for ISPH without shifting, ISPH with shifting, and SISPH scheme at $t = 1.2s$. ISPH without shifting is unstable, whereas shifting, and SISPH shows uniform distribution.

3.7.1.2 Change in the form of pressure gradient

Here we observe the effect of the two ways of computing the pressure gradient. The tolerance used for iteration is $\epsilon = 10^{-2}$. We also note that the "symm" form of pressure gradient works well even without the use of the GTVF formulation, but the "asymm" pressure gradient does not work without the addition of the GTVF. However, the "asymm" pressure gradient with the GTVF produces very good results, as can be seen in fig. 3.3. These result are better than the "symm" form for the pressure and decay rates with and without an initial perturbation. Hence for the Taylor-Green vortex we use the "asymm" form to compute pressure gradient.



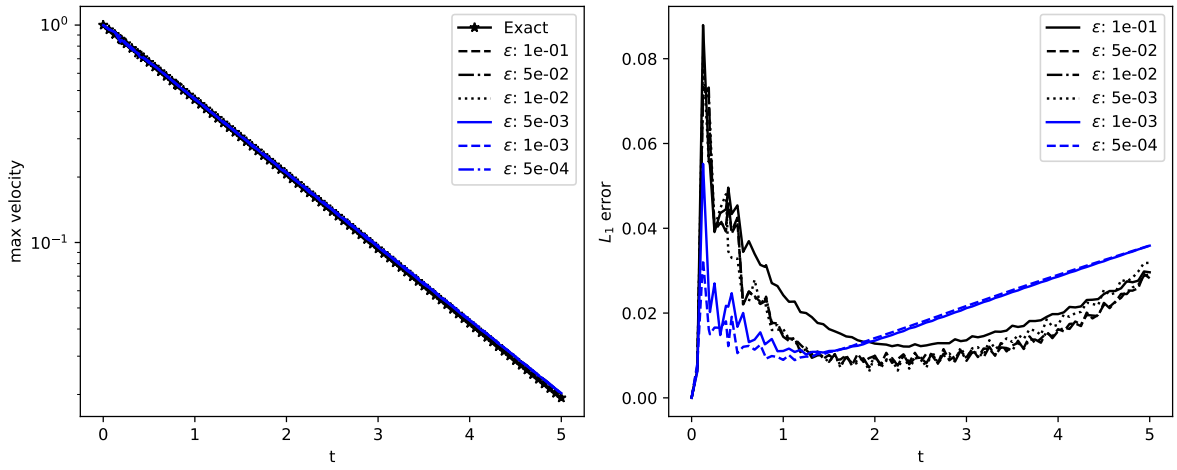
(a) Maximum velocity decay vs time.

(b) p_{L_1} vs time.

Figure 3.3: The use of "symm" form of pressure gradient vs "asymm" form of pressure gradient, in this simulation "asymm" is better than "symm".

3.7.1.3 Effect of tolerance

We study the variation in the iteration tolerance and its effect on accuracy for the SISPH scheme. We vary the tolerance in the range $\epsilon = [0.1, 0.05, 10^{-2}, 5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}]$. The "asymm" pressure gradient form with a 100×100 grid of particles is used. As can be seen in fig. 3.4, we see that the results are the same even as we vary the tolerance showing that it is not necessary to use very low tolerance when the spatial resolution does not demand it.



(a) Maximum velocity decay vs time.

(b) L_1 error of velocity magnitude vs t.

Figure 3.4: Change in tolerance, ϵ, for Taylor-Green vortex simulated for t = 5s using SISPH formulation.

3.7.1.4 Change in Re with varying resolution

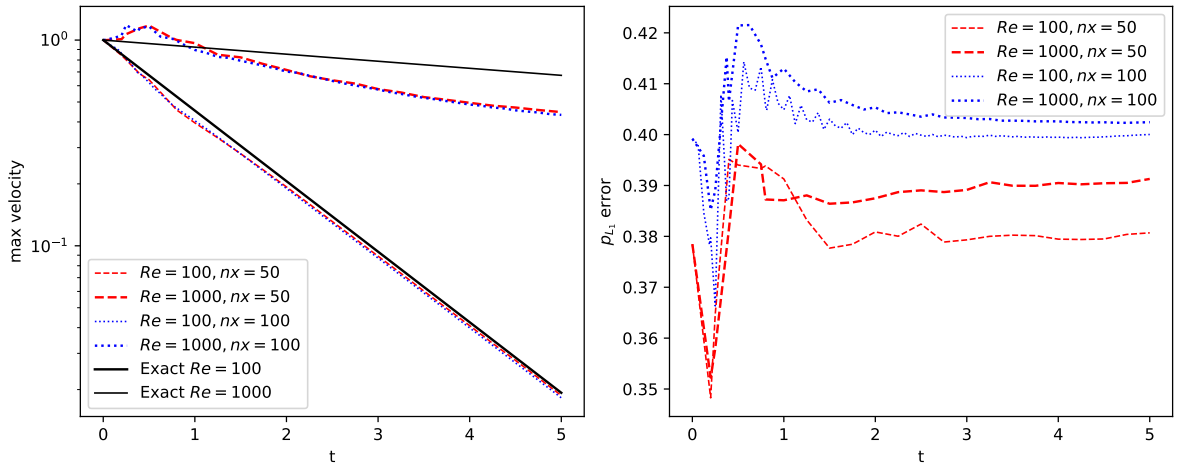
We next compare the Taylor-Green vortex with different Reynolds numbers, $Re = 100, 1000$, with different resolutions of 50×50 and 100×100. The tolerance is, $\epsilon = 10^{-2}$. "asymm" pressure gradient is used here. Figure 3.5 shows the results. These indicate that the scheme does work well for higher Reynolds numbers.

3.7.1.5 Comparison with other schemes

Here we compare with other established schemes such as WCSPH, IISPH and EDAC.All the schemes have a perturbed initial condition where the particles are displaced by a maximum of $\Delta x/5$ randomly about their original position with a uniform distribution. All the schemes use the same initial particle distribution. We use a tolerance of $\epsilon = 10^{-2}$ for the SISPH scheme, "asymm" form of pressure gradient with a 100×100 grid of initial particles.

In fig. 3.6a we note that EDAC and SISPH formulation matches the exact decay rate, and in the fig. 3.6b the errors in pressure, p_{L_1}, are much better compared to WCSPH and IISPH.

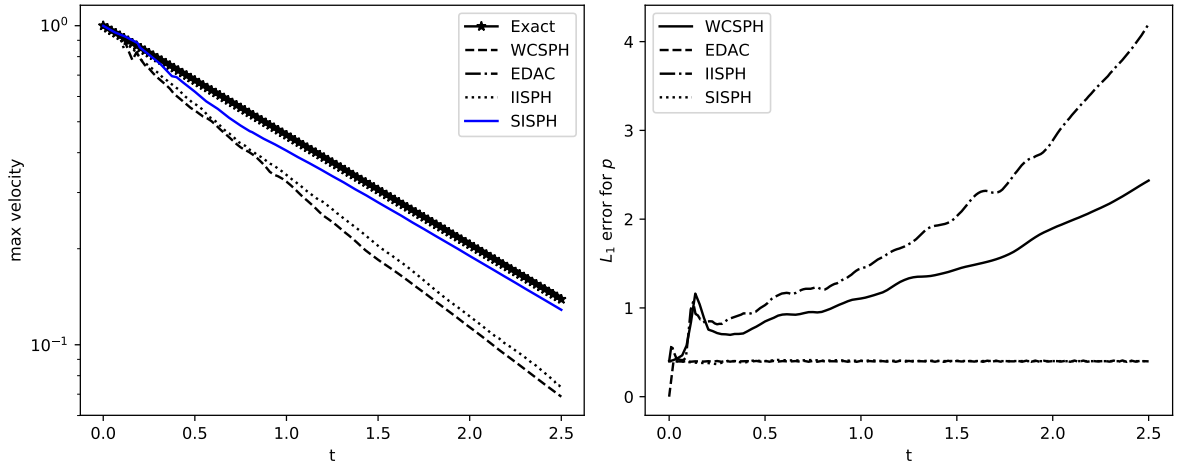
In conclusion SISPH with a tolerance of $\epsilon = 10^{-2}$ works well with the Taylor-Green vortex and hence we choose this for simulating the subsequent problems.



(a) Maximum velocity vs time.

(b) p_{L_1} with time.

Figure 3.5: Taylor-Green vortex simulated for t = 5s, with two different Reynolds numbers $Re = [100, 1000]$ and initial particle distribution of 50×50 and 100×100, using SISPH with "asymm" pressure gradient form.



(a) Maximum velocity decay vs time.

(b) p_{L_1} vs time.

Figure 3.6: Comparison of SISPH using "asymm" form of pressure gradient with WCSPH, EDAC and IISPH schemes. The initial configuration is perturbed by a uniformly distributed random number scaled by $0.2\Delta x$ for all the schemes and simulated for t = 2.5s.

3.7.2 Comparison with EISPH

Here we compare the Explicit ISPH (EISPH) (Barcarolo et al. 2014b; Hosseini et al. 2007; Nomeritae et al. 2016; Rafiee et al. 2009) with SIPSH. In EISPH, the PPE is solved with an explicit equation which is equivalent to one single iteration of the Jacobi iterations in each time step. In SISPH the PPE is solved iteratively to reach a desired tolerance per every time step. This is important in the case of high Reynolds number simulations. We compare the results using both the methods for the Taylor-Green vortex at $Re = 10000$. We perturb the initial particle positions by a maximum of $\Delta x/10$ without changing the mass or density.

Figure 3.7 shows the particle plots of EISPH vs SISPH where the color indicates velocity. This shows that the using EISPH approach leads to an inaccurate velocity. This shows the importance of multiple iteration per time step for high Reynolds number flows.

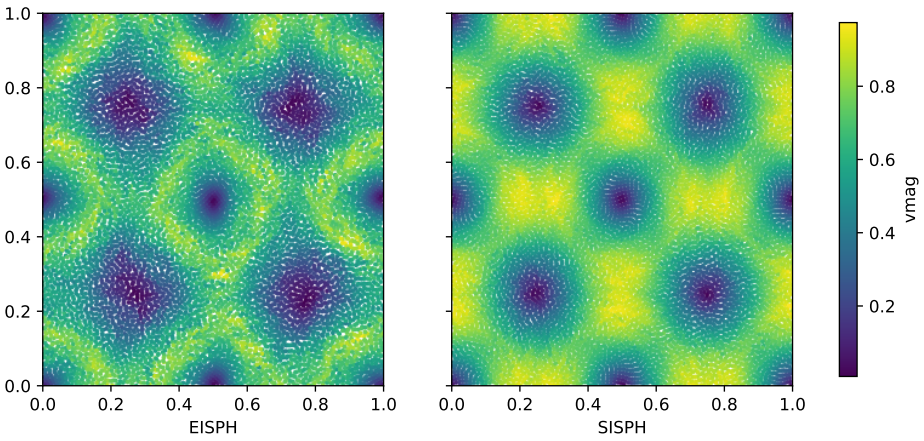


Figure 3.7: Comparison of EISPH with SISPH for $Re = 10000$, using 100×100 initial particle distribution. Particles are perturbed from their initial position by a maximum of $\Delta x/10$.

3.7.2.1 Performance of SISPH

Here we compare the performance of SISPH scheme with ISPH scheme on a CPU for a single-core and show the performance of SISPH on multi-cores. Figure 3.8 shows the time taken versus the number of particles, N, for the different cases. When N is large the SISPH method is an order of magnitude faster than the matrix based ISPH method. Figure 3.9 compares the scale up of SISPH with ISPH scheme on single core, the single core performance of SISPH is increase to

4 times, and the multi-core (with 4 cores) is nearly 12 times as fast as single core ISPH for large N. The matrix ISPH cannot be run on GPUs without a significant amount of programming effort whereas SISPH can be executed on the GPUs very easily as demonstrated for a 3D problem later.

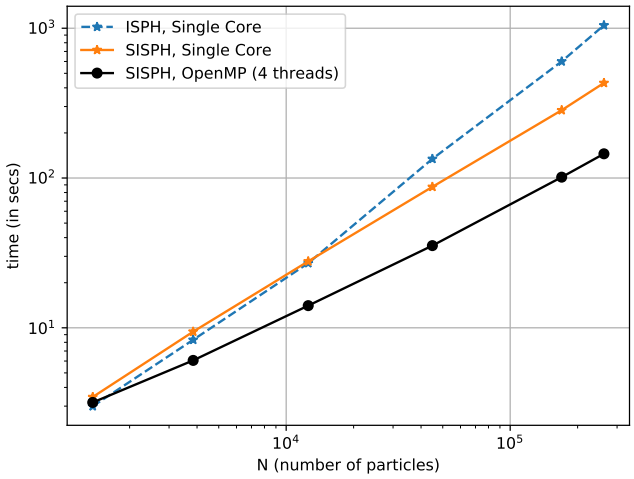


Figure 3.8: Log-log plot showing time taken vs number of particles for SISPH on single and multi-core CPUs, and matrix based ISPH on single-core CPU.

3.7.3 Lid-driven Cavity

The next problem considered is the classical lid-driven cavity problem. This problem allows us to test the solid wall boundary condition and the ability of the new scheme to handle the solid wall boundaries.

The problem involves a unit square vessel filled with incompressible fluid. The upper boundary over the fluid is made to move rightward with a unit speed, U. The other walls are treated as no-slip walls. For this problem the Reynolds number is defined as $Re = UL/\nu$, where L is the length of a side. We simulate the problem at $Re = 100$ with two grid sizes, 50×50 and 100×100. The results are compared with those of Ghia et al. (1982) where the velocity profile along a line passing horizontally and vertically through the center is shown after the simulation has stabilized. We also compare the results with those from the TVF (Adami et al. 2013) scheme. We use the parameter $h/dx = 1.0$ and employ the quintic spline kernel, the tolerance for SISPH

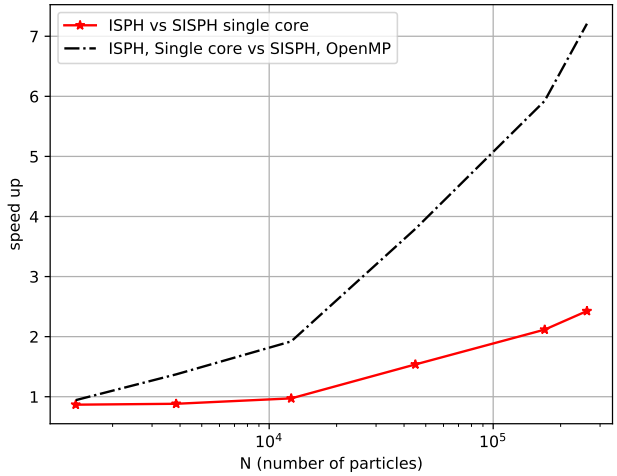


Figure 3.9: Performance speed up observed for SISPH scheme on single and multi-core with ISPH on single-core.

is, $\epsilon = 10^{-2}$, and since this is an internal flow the background pressure in GTVF formulation is fixed to a constant reference pressure which is $2\max(p_i)$ after the first iteration. The "symm" form of pressure gradient is used. Figure 3.10, plots the distribution of u and v along the center-line along the horizontal and vertical. The results are in very good agreement. In this simulation it is observed that, either use of "symm" form of pressure gradient or the "asymm" form doesn't affect the solution significantly, but the "symm" form works well even when no GTVF is used whereas the "asymm" form fails to work without the GTVF.

Due to the larger time steps the ISPH method allows, the new scheme is about 5 times faster than the TVF simulation. The $Re = 100$ case with 100×100 fluid particles takes only around 87 seconds to execute for 10 seconds of simulation time (on a quad core Intel i5–7400) with the new scheme. With the TVF scheme the same problem takes about 433 seconds. This shows that the new method works well and is much more efficient.

We next compare the results for $Re = 10000$. The velocity profiles show a good agreement with Ghia et al. (1982). It should however be noted that the simulation of Barcarolo (2013) shows voiding with $Re = 3200$. In our case the GTVF and an accurate pressure leads to very good results. Figure 3.12, plots the velocity magnitude of the cavity problem at $Re = 10000$. It highlights the difference between the boundary condition that is applied to the intermediate velocity ($\mathbf{u}^*$) when solving the PPE. In the plot on the left, the boundary condition on $\mathbf{u}^*$ is set to

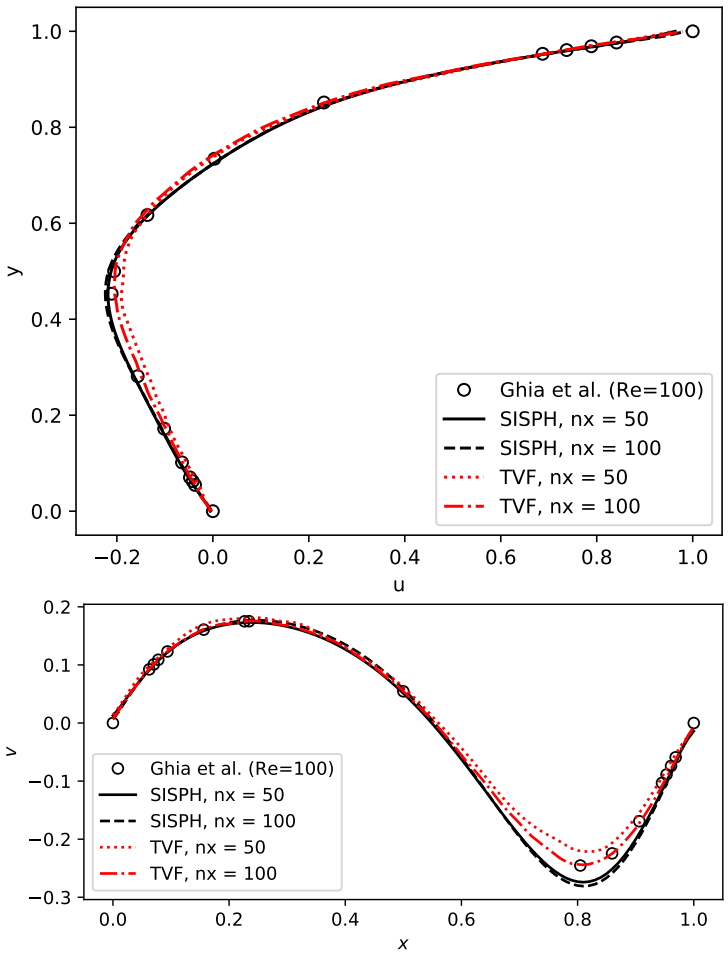


Figure 3.10: Velocity profiles u vs. y and v vs. x for the lid-driven cavity problem at $Re = 100$. Two particle discretizations of 50×50 and 100×100 are shown. We compare the results with those of Ghia et al. (1982) and with the TVF scheme.

ensure a no-slip wall, whereas, on the right, the boundary condition is set to ensure a slip wall. The slip wall boundary condition on the intermediate velocity reduces the noise introduced in the velocity field due to the unnecessary divergence created at the wall.

3.7.4 Evolution of a Square Patch

The simulation of an initially square patch is a free-surface benchmark introduced by Colagrossi (2005). Here, an initially square patch of fluid having length L is given the following initial

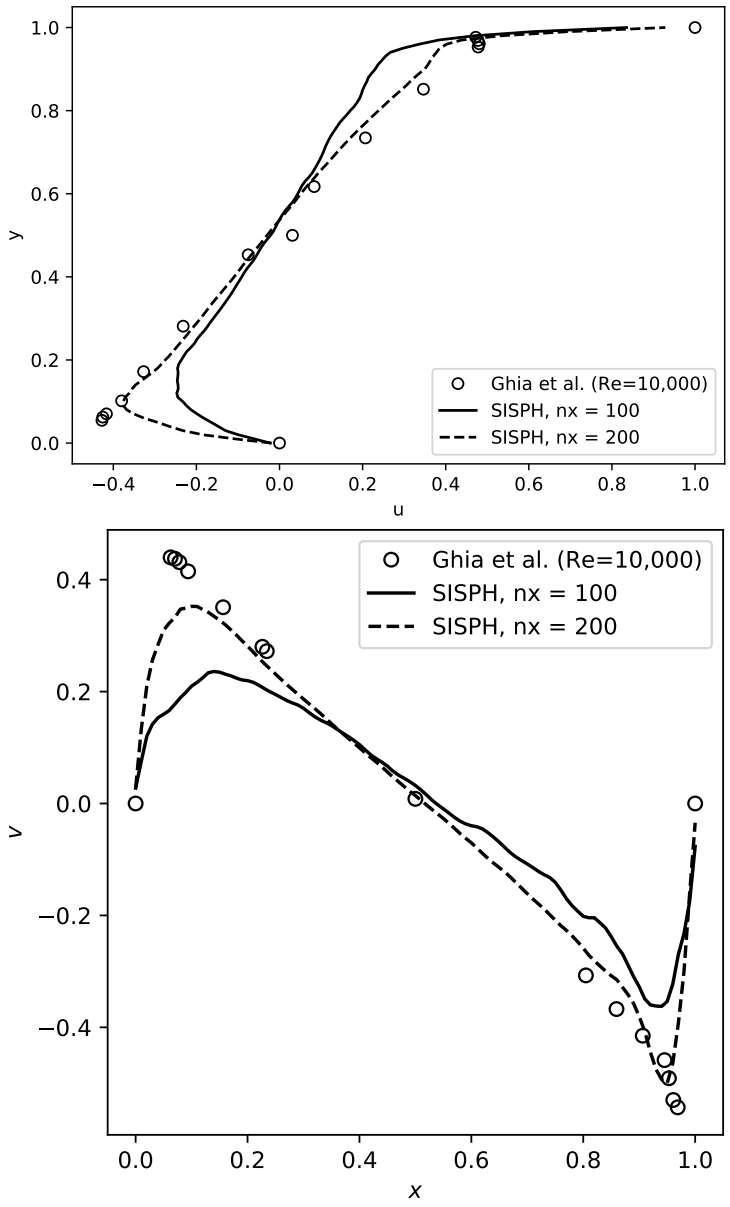


Figure 3.11: Velocity profiles u vs. y and v vs. x for the lid-driven cavity problem at $Re = 10,000$. Two particle discretizations of 100×100 and 200×200 are shown. We compare the results with those of Ghia et al. (1982).

conditions,

$$\begin{aligned} u_0(x, y) &= \omega y \\ v_0(x, y) &= -\omega x \end{aligned} \tag{3.47}$$

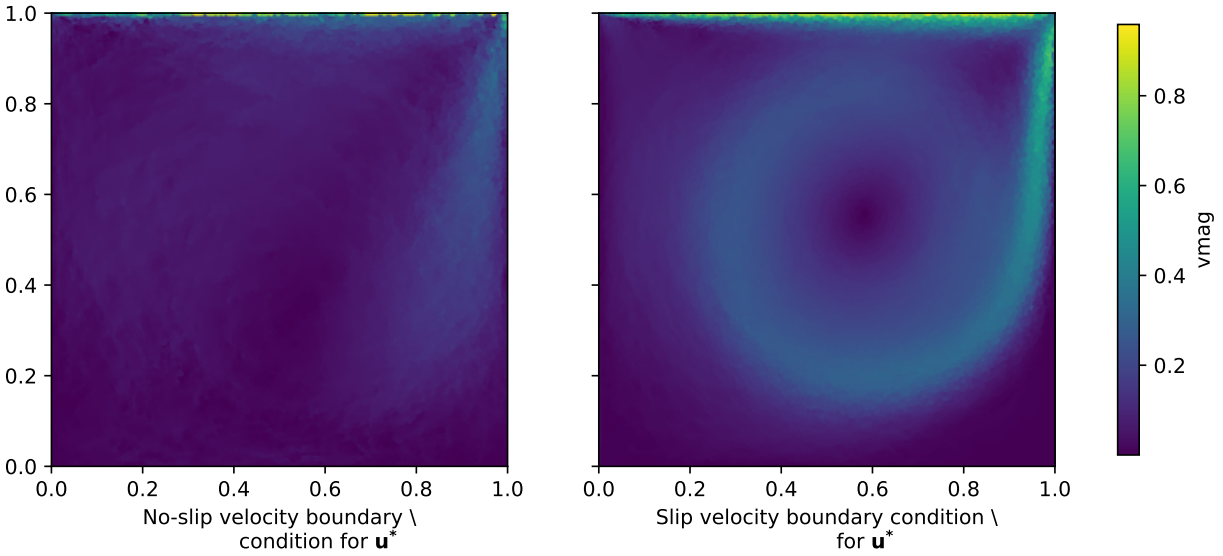


Figure 3.12: Particle plots indicating the velocity magnitude for the lid-driven cavity problem at $Re = 10000$. Two different boundary conditions are used in the plots. On the left the no-slip wall boundary condition is used on the intermediate velocity while solving the PPE. On the right we use the slip velocity boundary condition.

$$p_0(x,y) = \rho \sum_{m}^{\infty}\sum_{n}^{\infty} -\frac{32\omega^2/(mn\pi^2)}{\left[\left(\frac{n\pi}{L}\right)^2 + \left(\frac{m\pi^2}{L}\right)^2\right]} \sin\left(\frac{m\pi x^*}{L}\right)\sin\left(\frac{n\pi y^*}{L}\right) m,n \in \mathbb{N}_{odd} \tag{3.48}$$

where $X^* = x + L/2$ and $y^* = y + L/2$.

This problem is simulated for 3 seconds using the new scheme, and the WCSPH scheme, for the discussion on the results of IISPH scheme see Ramachandran et al. (2021a). The quintic spline kernel with $h/\Delta x = 1.3$ is used for all schemes. An artificial viscosity coefficient of $\alpha = 0.15$ is used for the WCSPH and the new scheme. The tolerance chosen for the new scheme is $\epsilon = 10^{-2}$. Initial particle distribution of 100×100 is used for all the schemes.

A noticeable difference is observed when using the different forms of pressure gradient, the "asymm" gradient of pressure simulates the problem significantly better than the other, where even though the core structure remains, the particles along the free surface are deviated further. We also observed that computing the density after each time step using summation density is very crucial for this problem, without it i.e., using a constant density through out makes the simulation unstable. A comparison with WCSPH scheme is made, and the effect of pressure gradient is shown in the fig. 3.13. It is clear that the new scheme with the "asymm" form of

pressure gradient performs very well.

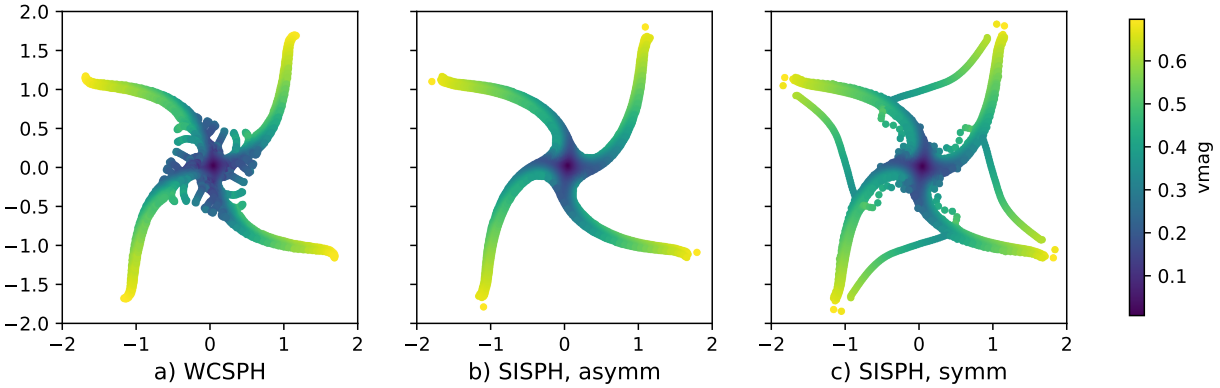


Figure 3.13: Distribution of particles at $t = 3s$ for the initially square patch, with a 100×100 initial particle, distribution. SISPH scheme is compared with WCSPH scheme. Effect of different forms of pressure gradient is shown. In the "symm" pressure gradient particles along the free surface are deviated further.

3.7.5 Dam-break in Two Dimensions

The dam-break problem in two dimensions is a classic problem involving free-surfaces and solid walls. We simulate the standard problem as described in (Lee et al. 2010). This involves a vessel of width 4m with a block of fluid of width 1m and height 2m on the left side. The block of water is released from rest and falls under the influence of gravity ($g = 9.81m/s$).

The problem is simulated with the proposed scheme using a quintic spline kernel. Artificial viscosity of $\alpha = 0.05$ is used for all the schemes. The fluid is not treated as viscous. The $h/\Delta x = 1.3$ used for all the schemes, the default particle spacing, $\Delta x = 0.01$, is used which results in 37k particles. The time step is fixed and chosen as $\Delta t = 0.125h/\sqrt{2gh_w}$, where g is the acceleration due to gravity and h_w is the initial height of the water which is 2m. The tolerance value for the new scheme is, $\epsilon = 10^{-2}$. The "symm" form of pressure gradient is used, although no significant difference is observed with either of the forms.

The particle distribution of the flow at different times is plotted in fig. 3.14. The results show that the scheme works very well. Figure 3.15 plots the toe of the breaking water versus time. The results are compared with the WCSPH scheme, EDAC scheme and the numerical simulations in Koshizuka et al. (1996). As can be seen, the results are in very good agreement.

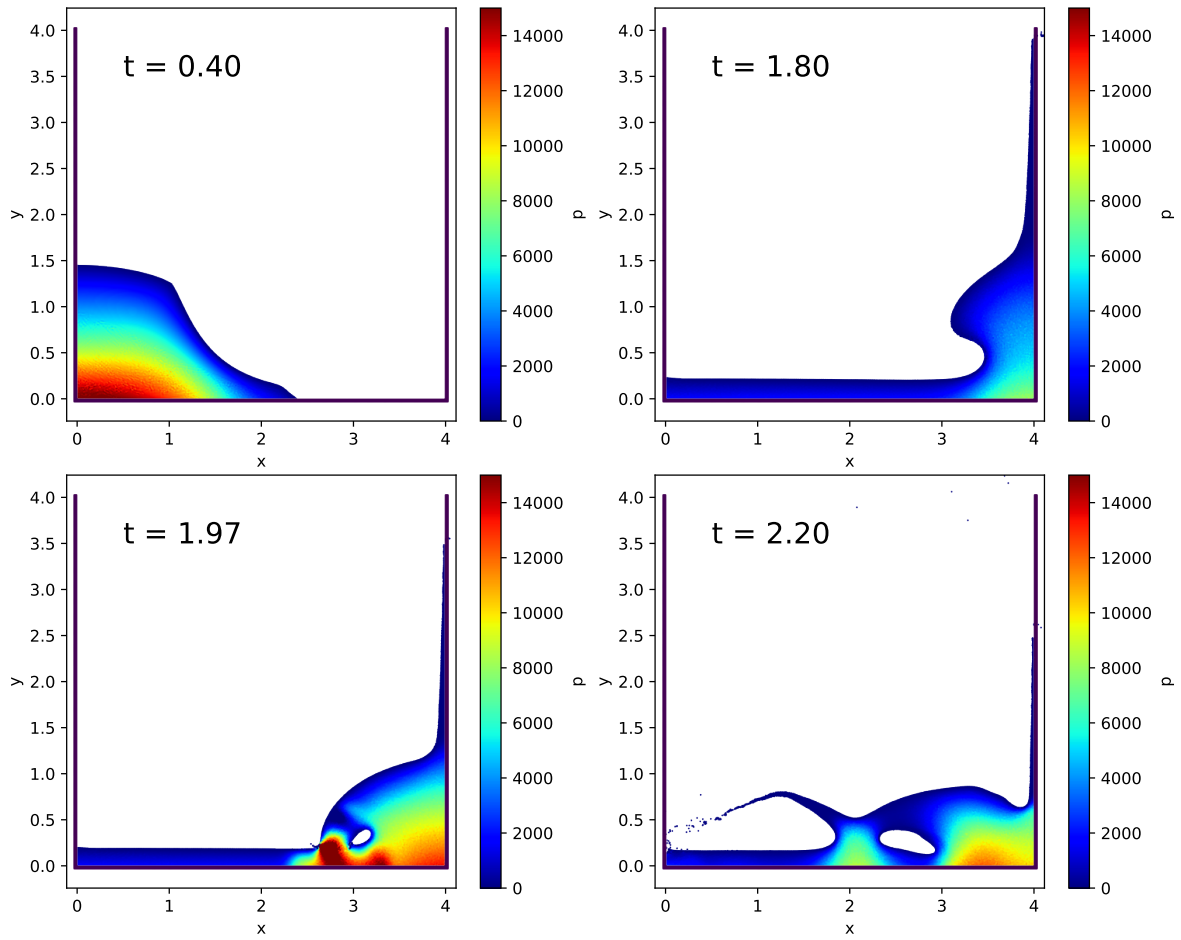


Figure 3.14: Particle plots of Dam-break in 2D showing pressure at various times using SISPH.

3.7.6 Dam-break in Three Dimensions

A three-dimensional dam-break is considered in order to show that the scheme works in three dimensions and demonstrates the performance of the scheme on a GPU. The problem considered is a small modification of the problem described in (Lee et al. 2010), wherein an obstacle is placed in the path of the flow. We do not use an obstacle in our example. A rectangular block of fluid having height $h_w = 0.55m$, width of $1m$ and length of $1.228m$ is at rest at one end of a container. The container is a long rectangular container of length $3.22m$ and open at the top. Four layers of boundary particles are used to enforce the no-penetration boundary condition. A quintic spline kernel is used with $h/\Delta x = 1.0$. The reference velocity for the flow is $v_{ref} = \sqrt{2gh_w}$, where $g = 9.81m/s^2$, and the tolerance is, $\epsilon = 0.01$. The particle spacing

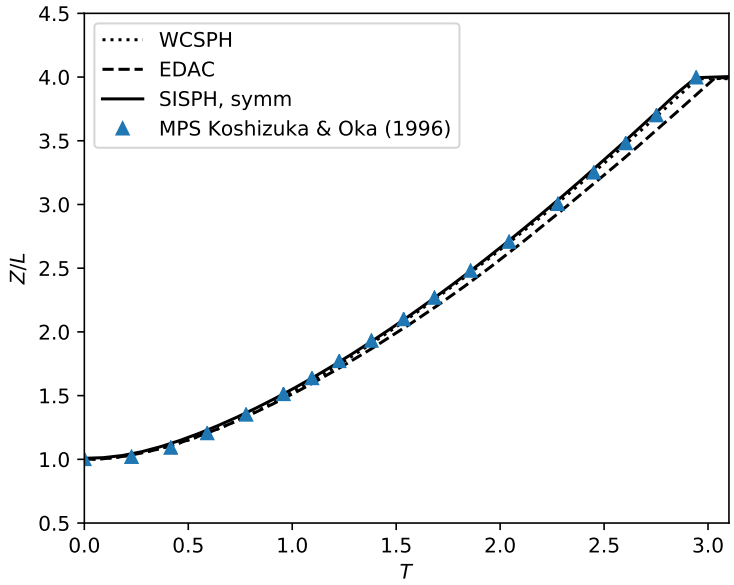


Figure 3.15: X-coordinate of the toe of the dam versus time as computed with the new scheme, WCSPH, EDAC and the simulation of (Koshizuka et al. 1996). Z is the distance of toe of the dam from the left wall and L is the initial width of the dam

is chosen as $\Delta x = 0.01m$, this results in around 664k fluid particles and around 627k solid wall particles. The time step is chosen to be $h/(8v_{ref})$. An artificial viscosity is used with the parameter $\alpha = 0.25$. The results at different times are shown in fig. 3.16. There is a large amount of splashing initially but the scheme produces good results and are similar to those of Chow et al. (2018).

The code is executed using OpenCL on an NVIDIA 1070 Ti processor (TDP of 180 watts) with single precision. It is also executed with double precision on a quad core Intel i5–7400 CPU (TDP of 65 watts) running at 3.0 Ghz. The user-code is identical as PySPH (Ramachandran et al. 2021b) automatically executes the code using OpenCL or OpenMP depending on the options passed. On the GPU, the execution for 3 seconds of simulation takes 4531 seconds and the same simulation on the CPU with OpenMP takes 40264 seconds suggesting that the GPU execution is around 8.9 times faster than the CPU (quad core) for this size of problem. The cost savings in terms of power usage for running on the GPU are around 3.21 times higher than for a purely CPU execution. Note that the cost estimate is based solely on the wattage of the individual devices (CPU and GPU) and not on the total wattage of the entire unit, which may include other electrical components.

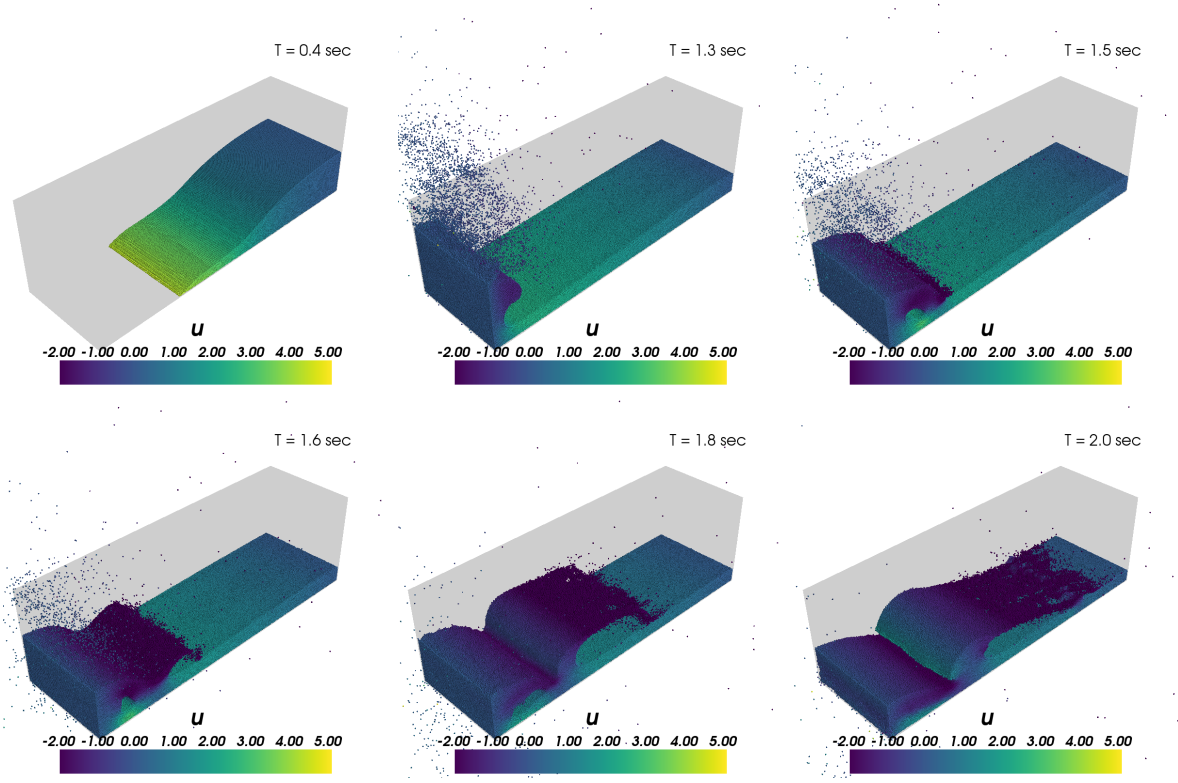


Figure 3.16: Dam-break in three dimensions using SISPH shown at times t = (0.4, 1.3, 1.5, 1.6, 1.8, 2.1)secs, color indicates u velocity.

3.7.6.1 Performance of SISPH

Here we show the performance of SISPH scheme on single-core, and multi-core CPU (Intel i5–7400) with double precision, and NVIDIA 1050Ti, 1070Ti GPUs with single precision. In fig. 3.17 the increase in number of particles, N, vs time taken by SISPH on various platforms is show, CPUs outperform GPUs when N is small as can be expected. For sufficiently large N, at around 100k particles, we can start to see the improved performance of the GPUs (albeit with single precision), and for 1M particles the time is reduced by an order of magnitude, also the "scale up" is linear. Figure 3.18 shows speed up of SISPH on multi-core and GPUs compared to that of single core CPU, while the performance on multi-core (with 4 cores), as expected, peaks around 4 times that of single core, performance of GPUs increases with number of particles and reaches the peak of around 16 times that of single CPU core for 1050Ti and around 32 times for 1070Ti, showing the increase in performance with the proposed scheme.

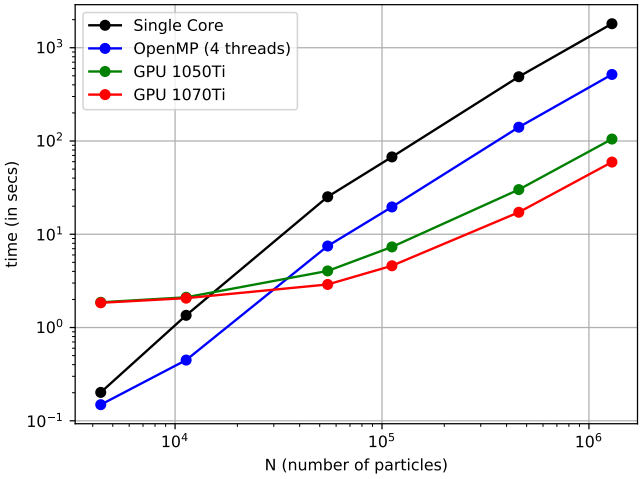


Figure 3.17: Log-log plot showing the performance of SISPH on different platforms, no of particles are shown on the x-axis and time taken is shown on the y-axis.

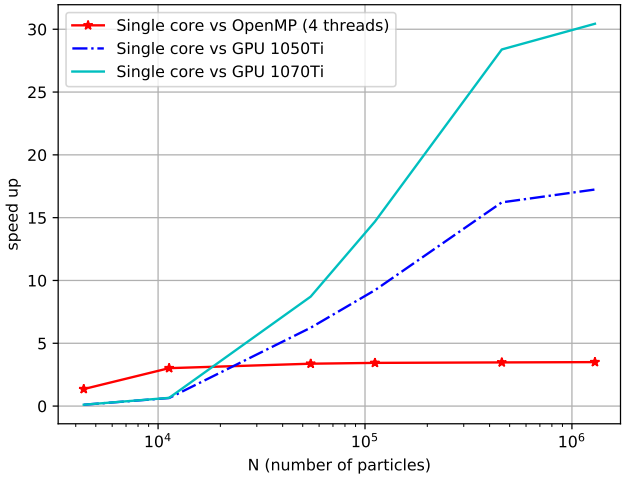


Figure 3.18: Plot showing the scale up of SISPH on multi-core, NVIDIA 1050Ti, and 1070Ti GPU vs performance on single core.

3.7.7 Flow Past a Circular Cylinder

As a final example, the flow past a circular cylinder is considered. A cylinder of diameter $D = 2m$ is considered, the problem setup is as shown in fig. 3.19. Note that the particles discretizing the

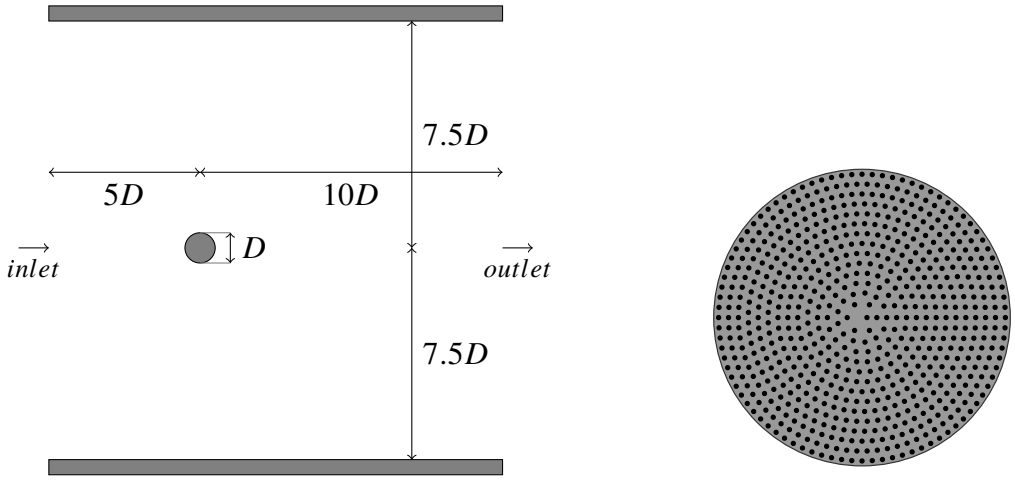
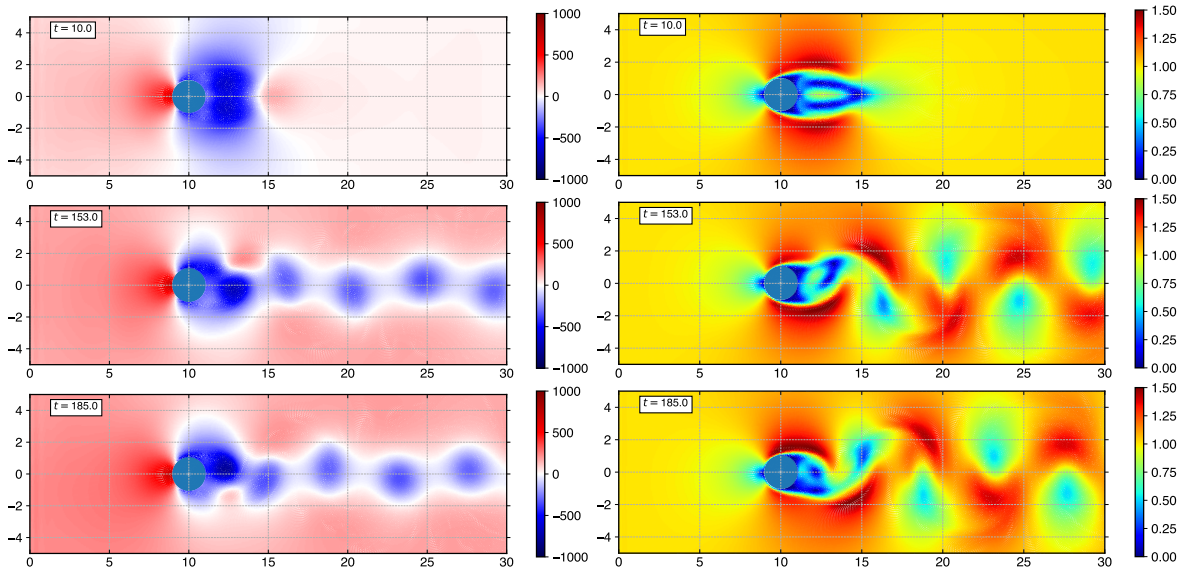


Figure 3.19: The figure on the left shows the setup for flow past cylinder, the right figure shows the particle distribution on the cylinder.

cylinder are placed at a distance of $\Delta x/2$ from the outer circumference in order to effectively simulate the correct location of the boundary of the cylinder. The cylinder is placed a distance of $5D$ from the inlet and the outlet is placed $10D$ from the center of the cylinder. The inlet is set to a uniform velocity, U, along the x-axis of 1m/s. The sides of the wind-tunnel are set to slip walls. These walls are placed at a distance of $7.5D$ from the center of the cylinder. The viscosity is set to ensure a Reynolds number, $Re = UD/\nu = 200$. No artificial viscosity is used. The flow is started impulsively and simulated for 200 seconds. A quintic spline kernel is used with $h/\Delta x = 1.2$. Boundary conditions are implemented as discussed in Sections 3.5 and 3.5.1. The convergence tolerance is set as $\epsilon = 0.01$. The particle spacing is chosen as $\Delta x = D/30$, which results in 200k fluid particles. The "symm" form is used for the calculation of pressure gradient. Since the resolution is low, the geometry curvature is captured by placing particles along the circumference of the cylinder such that the volume occupied by the particles is approximately constant as done in (Negi et al. 2019). For this case we encountered particle voiding when we use the TVF reference pressure of $2\max(p_i)$. This was resolved by choosing $p_{\text{ref}} = \rho c^2$, where c is $10|\mathbf{U}|_{\max}$.

In fig. 3.16, we plot the velocity and pressure contour by taking the snapshot at times 10s, 153s and 185s. The pressure and velocity contour shows an excellent match with the result presented in (Marrone et al. 2013a; Tafuni et al. 2018). The pressure and velocity variations have much less noise than the WCSPH schemes even with a low resolution of particles as



(a) Pressure contour

(b) Velocity contour

Figure 3.20: Plots showing the pressure and velocity contours at times = (10s, 153s & 185s), as simulated by SISPH while employing "symm" form of pressure gradient eq. (3.7). A $\Delta x = D/30$ is used here resulting in 200k fluid particles.

compared to (Negi et al. 2019).

The drag and lift coefficients, c_d, c_l are plotted as a function of time in fig. 3.21. It must be noted that unlike WCSPH schemes, the data presented has not gone through any filtering to remove high frequency oscillations. In order to calculate the forces on the cylinder, we compute acceleration on the solid due to the pressure and velocity gradients using the following equation,

$$\frac{\mathbf{f}_{\text{solid}}}{m_{\text{solid}}} = -\nabla p + \nu \nabla \cdot (\nabla \mathbf{u}) \tag{3.49}$$

where $\mathbf{f}_{\text{solid}}$ is the force on the cylinder particles, using SPH discretization (Adami et al. 2013; Ramachandran et al. 2019), the above equation is written as,

$$\mathbf{f}_{i,\text{solid}} = \sum_j \left(V_i^2 + V_j^2\right) \left[-\tilde{p}_{ij} \nabla W_{ij} + \tilde{\eta}_{ij} \frac{\mathbf{u}_{ij}}{(r_{ij}^2 + \eta h_{ij}^2)} \nabla W_{ij} \cdot \mathbf{r}_{ij}\right] \tag{3.50}$$

where,

$$\tilde{p}_{ij} = \frac{\rho_j p_i + \rho_i p_j}{\rho_i + \rho_j}, \tag{3.51}$$

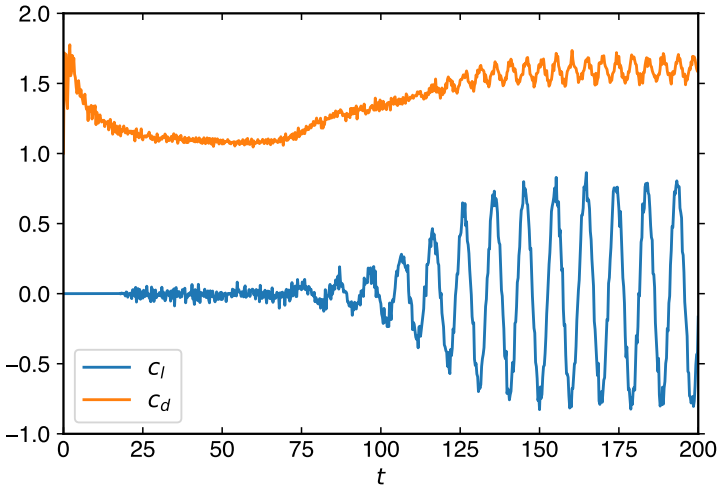


Figure 3.21: The coefficients of lift, c_l, and drag, c_d, vs time are plotted here. After the initial fluctuation, the average coefficient of drag, $c_d = 1.609$ is observed, the maximum lift coefficient is, $c_l = 0.804$.

$$\tilde{\eta}_{ij} = \frac{2\eta_i\eta_j}{\eta_i + \eta_j}, \tag{3.52}$$

$$\mathbf{u}_{ij} = \mathbf{u}_{w\,i} - \mathbf{u}_j \tag{3.53}$$

where $\mathbf{u}_{w\,i}$ is the velocity of the solid wall particles as obtained from eq. (3.39), $\eta_i = \rho_i \nu_i$.

We obtain an average coefficient of drag, $c_d = 1.609$ (once the vortex shedding has been established) and the maximum coefficient of lift, $c_l = 0.804$. In the fig. 3.21, c_d and c_l show clear oscillations without any change in the amplitude, which is consistent with the results of others. With this we have demonstrated the suitability and robustness of the proposed scheme for a variety of problems.

3.8 Consolidated Overview of the SPH Methods Proposed

In the earlier sections, a numerical evaluation of SISPH was performed. Here, we comment on the various techniques that were incorporated into the base scheme of matrix-free ISPH. We have discretized the Navier-Stokes equations using the Generalized Transport Velocity Formulation. The transport velocity is obtained by shifting the particles using eq. (3.14) (Lind et al. 2012). We recommend in both the internal, and the external flows with free-surfaces, the “asymm” form

of the pressure gradient (eq. (3.8)). This is more stable and less diffusive as observed from the Taylor-Green simulation. Also, in the problems with moderate Reynolds number of around 10,000, "asymm" form shows better stability. In the cases with viscosity we do not use any artificial viscosity but if the flow is inviscid we add an artificial viscosity with the parameter α, in the range of $0.05 - 0.25$.

In addition, a comparison was made between SISPH and other SPH methods. When the Taylor-Green problem was evaluated, the WCSPH based scheme without any shifting exhibited significant numerical dissipation in comparison to the SISPH scheme. The time-step for weakly-compressible schemes is restricted by the inverse of the absolute sum of artificial speed of sound and maximum velocity, $|\mathbf{u}_{\max} + c_s|$, while for SISPH it is only restricted by the maximum velocity, $|\mathbf{u}_{\max}|$. Explicit ISPH without particle shifting and no iterative formulation was unstable at moderate Reynolds number, but SISPH showed a reasonable match for a Reynolds number of 10,000 in the lid-driven cavity case. However, the Taylor-Green problem at *Re* 10,000 exhibited moderate dissipation in the maximum velocity. The expected distribution of pressure was obtained for the two-dimensional dam-break case and the flow past cylinder.

3.9 Summary

A simple, iterative, incompressible SPH scheme is developed that is based on the original projection formulation of Cummins et al. (1999). The method is matrix-free, fast, and suitable for execution on GPUs. The formulation is simple and easy to implement. We introduce a novel technique to ensure a homogeneous distribution of particles. In addition we introduce a modified solid wall boundary condition and show how we can implement simple inlet and outlet boundary conditions (refer to Muta et al. (2020a) for the flow past circular cylinder problem). We demonstrate the accuracy and efficiency of the new scheme with a suite of benchmark problems in two and three dimensions involving internal and external flows.

4 Dual-Time Smoothed Particle Hydrodynamics

One person's clear mental image is another person's intimidation.

— William Thurston, *On proof and progress in mathematics (1994)*

4.1 Introduction

In this chapter we develop a new scheme for weakly-compressible fluid flows. Our work takes inspiration from the Artificial Compressiblility-based Incompressible SPH (ACISPH) scheme proposed by Rouzbahani et al. (2017). Our scheme uses a different formulation that is also very efficient. The original scheme (Rouzbahani et al. 2017) was not noted in particular for its efficiency. We propose an original derivation and suggest many improvements that make the proposed scheme efficient. The performance is significantly better than that of the traditional WCSPH schemes and comparable to that of the ISPH schemes without sacrificing any accuracy or being unduly hard to implement. Our approach employs the classic artificial compressibility of Chorin (1967) in a dual-time stepped framework. We call the resulting scheme, DTSPH for dual-time stepped SPH.

The new scheme is designed to be implemented as an extension of the classic weakly-compressible schemes. The advantage of this is that these are relatively easy to implement, boundary conditions may be easily enforced, and there are several well-established schemes that may be used. Although we have not done so in this work, it is important to note that this

approach may also be employed for solid mechanics problems where the artificial speed of sound is usually very large.

The new scheme can be adapted to any WCSPH formulation which uses a density or pressure evolution equation based on a continuity equation. We demonstrate the scheme with the EDAC scheme (Ramachandran et al. 2019). We note that it can be easily applied to other schemes like the δ-SPH. We show how our scheme can be used to obtain steady state solutions, although this is not particular to the new scheme and can be easily performed for a variety of other schemes. Obtaining steady state solutions in the context of SPH simulations is useful in different contexts. For example, in the case of the impulsively started flow past a complex geometry, an initial potential flow solution is useful and this may be easily obtained using this approach.

The density-based dual-time stepping method proposed by Fatehi et al. (2019) is in principle similar to the present work, but it does not demonstrate any significant performance advantage over the traditional WCSPH schemes. Moreover, they do not demonstrate their method with any free-surface problems. Our proposed implementation and method is however significantly faster than the WCSPH scheme and has been demonstrated to work well for free-surface and three-dimensional problems. The method of Zhang et al. (2020) is very different from the method proposed here and have higher memory requirements. The present method demonstrates much improved performance and does not involve a significant memory penalty.

We provide a new formulation as compared to the work of (Rouzbahani et al. 2017). We explore several important details for the implementation of the scheme, and more importantly provide a high-performance, open source implementation of the scheme. Our implementation uses the open source PySPH framework (Ramachandran et al. 2021b) and all the code related to the work done is available at `https://gitlab.com/pypr/dtsph`. In order to facilitate reproducible research, the work is completely reproducible and every figure in this chapter is automatically generated using `automan` (Ramachandran 2018).

In the next section we discuss the proposed DTSPH scheme in a general setting and then discuss the SPH discretization. We briefly discuss the stability requirements of the scheme. We show how the resulting scheme is efficient and then proceed to simulate various standard benchmark problems. We perform comparisons with the TVF (Adami et al. 2013), the δ-SPH scheme (Antuono et al. 2010), and the EDAC scheme (Ramachandran et al. 2019) where relevant to demonstrate the accuracy and efficiency of the proposed scheme.

4.2 The Dual-Time SPH Method

In dual-time stepping schemes, a new time dimension called the "dual-time", denoted by the variable τ, is introduced. We have the following important considerations to keep in mind. If $\mathbf{r}$ is the position vector of a particle, then the *real* velocity of the particle is defined as, $\mathbf{u} = \frac{d\mathbf{r}}{dt}$. On the other hand, if the particle were to move in pseudo-time, we define the velocity in pseudo-time as $\hat{\mathbf{u}} = \frac{d\mathbf{r}}{d\tau}$.

If we consider a property of the particle, p, then we can define a material derivative in pseudo-time as,

$$\frac{dp}{d\tau} = \frac{\partial p}{\partial \tau} + \hat{\mathbf{u}} \cdot \text{grad}(p). \tag{4.1}$$

The ACISPH formulation (Rouzbahani et al. 2017) uses a non-dimensionalized form of the equation,

$$\frac{1}{\beta^2}\frac{\partial p^*}{\partial \tau^*} + \text{div}(\mathbf{u}^*) = 0. \tag{4.2}$$

Here we use a star for the non-dimensional terms. By assuming that $p = (\rho - \rho_0)c_s^2$ where c_s is an artificial sound speed and ρ_0 is a reference density, the above can be written in a dimensional form as,

$$\frac{\partial p}{\partial \tau} = -\rho c_s^2 \text{div}(\mathbf{u}). \tag{4.3}$$

By rewriting the non-dimensional form in eq. (4.2) with suitable dimensional quantities, it is easy to see that $\beta = 1/M$ where $M = u_{\text{ref}}/c_s$ is the Mach number of the reference speed of the flow, u_{ref}.

If we move the particles in pseudo-time, we introduce a material derivative to get,

$$\frac{dp}{d\tau} = \hat{\mathbf{u}} \cdot \text{grad}(p) - \rho c_s^2 \text{div}(\mathbf{u}). \tag{4.4}$$

With the EDAC formulation (Ramachandran et al. 2019), the pressure evolution equation also includes a diffusive term and we get,

$$\frac{dp}{d\tau} = \hat{\mathbf{u}} \cdot \text{grad}(p) - \rho c_s^2 \text{div}(\mathbf{u}) + \nu_e \nabla^2 p, \tag{4.5}$$

where ν_e is an artificial viscosity parameter which is typically chosen as,

$$\nu_e = \frac{hc}{16}. \tag{4.6}$$

Here h is the smoothing length. In the above, we have assumed that the particles move in pseudo-time. In later sections, we show that it is computationally efficient to not move the particles in pseudo-time. This results in a simpler equation for the pressure evolution without a time derivative as,

$$\frac{\partial p}{\partial \tau} = -\rho c_s^2 \text{div}(\mathbf{u}) + \nu_e \nabla^2 p. \tag{4.7}$$

The momentum equation can be written similarly by adding a time derivative of velocity in pseudo-time,

$$\frac{\partial \mathbf{u}}{\partial \tau} + \frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\text{grad}(p) + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{4.8}$$

where ν is the kinematic viscosity of the fluid and $\mathbf{f}$ is the body force. Again, if we choose to move the particles in pseudo-time, we can write this in terms of a material derivative in pseudo-time as,

$$\frac{d\mathbf{u}}{d\tau} + \frac{d\mathbf{u}}{dt} = \hat{\mathbf{u}} \cdot \text{grad}(\mathbf{u}) - \frac{1}{\rho}\text{grad}(p) + \nu \nabla^2 \mathbf{u} + \mathbf{f}. \tag{4.9}$$

Note the key difference here from what is proposed in (Rouzbahani et al. 2017) is that they have used $\mathbf{u}$ where they should have only used $\hat{\mathbf{u}}$.

There are two possible approaches we can choose for implementation:

- Move particles in real time and pseudo-time and use the eqs. (4.4) and (4.9). Note that when using the EDAC formulation we would add the pressure diffusion term as shown in the right hand side of eq. (4.5).
- Move the particles only in real time and use eq. (4.3) (or eq. (4.7) in the case of the EDAC scheme) and eq. (4.8).

We discuss these two approaches next.

4.2.1 Moving Particles in Pseudo-Time

In this section we show how the above equations are integrated in real time and pseudo-time. The following equations apply to each particle, i. We suppress the subscript i in the following to simplify the notation. We use the index k to denote pseudo-time iterations and n for the real time. Before iterating in pseudo time the particles are updated to a guessed new state ($k = 0$) for the next real time ($n + 1$) using,

$$\mathbf{u}^{k=0} = \mathbf{u}^n + \Delta t \left(\frac{d\mathbf{u}}{dt}\right)^n, \tag{4.10}$$

$$\mathbf{r}^{k=0} = \mathbf{r}^n + \Delta t \mathbf{u}^n + \frac{\Delta t^2}{2}\left(\frac{d\mathbf{u}}{dt}\right)^n, \tag{4.11}$$

where $(\frac{d\mathbf{u}}{dt})^n$ is given by the momentum eq. (4.8) without the partial derivative of velocity in pseudo time (i.e. considering $\frac{\partial \mathbf{u}}{\partial \tau} = 0$). Then the integration in pseudo time proceeds in the following fashion, with $\hat{\mathbf{u}}^{k=0} = 0$ as the starting value,

$$\mathbf{r}^{k+1/2} = \mathbf{r}^k + \frac{1}{2}\Delta\tau\hat{\mathbf{u}}^k, \tag{4.12}$$

$$p^{k+1/2} = p^k + \frac{1}{2}\Delta\tau\left(\frac{dp}{d\tau}\right)^k, \tag{4.13}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\tau\left(\frac{d\mathbf{u}}{d\tau}\right)^{k+1/2}, \tag{4.14}$$

$$\mathbf{r}^{k+1} = \mathbf{r}^{k+1/2} + \frac{1}{2}\Delta\tau\hat{\mathbf{u}}^{k+1}, \tag{4.15}$$

$$p^{k+1} = p^{k+1/2} + \frac{1}{2}\Delta\tau\left(\frac{dp}{d\tau}\right)^{k+1/2}. \tag{4.16}$$

In addition to these we have,

$$\hat{\mathbf{u}}^{k+1} = \left(\frac{d\mathbf{u}}{d\tau}\right)^{k+1/2}\Delta t. \tag{4.17}$$

This is to ensure consistency of the motion. A detailed proof for eq. (4.17) is provided in the appendix Section 7.1.

We need an expression for the term $\frac{d\mathbf{u}}{dt}$ in the momentum eq. (4.9). We obtain this using an implicit three-point backward difference scheme to discretize the real-time derivative,

$$\left(\frac{d\mathbf{u}}{dt}\right)^{n+1} = \frac{3\mathbf{u}^{n+1} - 4\mathbf{u}^n + \mathbf{u}^{n-1}}{2\Delta t} + O(\Delta t)^2. \tag{4.18}$$

Substitute $\mathbf{u}^{k+1}$ instead of $\mathbf{u}^{n+1}$ in eq. (4.18) and add and subtract the term $3\mathbf{u}^k$ in the numerator and rearrange terms to get,

$$\left(\frac{d\mathbf{u}}{dt}\right)^{n+1} = \frac{3\Delta\tau}{2\Delta t}\left(\frac{d\mathbf{u}}{d\tau}\right)^{k+1/2} + \frac{3\mathbf{u}^k - 4\mathbf{u}^n + \mathbf{u}^{n-1}}{2\Delta t} + O(\Delta t)^2. \tag{4.19}$$

If we use eq. (4.9), we can rewrite the above as,

$$\begin{aligned}\left(\frac{d\mathbf{u}}{d\tau}\right)^{k+1/2} \approx \Bigg\{&\hat{\mathbf{u}}\cdot\mathrm{grad}(\mathbf{u}) - \frac{1}{\rho}\mathrm{grad}(p) + \nu\nabla^2\mathbf{u} + \mathbf{f} \\ &- \frac{(3\mathbf{u}^k - 4\mathbf{u}^n + \mathbf{u}^{n-1})}{2\Delta t}\Bigg\}\left(\frac{2\Delta t}{2\Delta t + 3\Delta\tau}\right).\end{aligned} \tag{4.20}$$

We may now discretize the right hand side using SPH and find the acceleration to the velocity in pseudo-time. This is done in Section 4.3.

4.2.2 Fixed Particles in Pseudo-Time

In this section we consider the case where the particles are not moved in pseudo-time. In this case, we can perform the integration as follows,

$$p^{k+1/2} = p^k + \frac{1}{2}\Delta\tau\left(\frac{\partial p}{\partial \tau}\right)^k, \tag{4.21}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\tau\left(\frac{\partial \mathbf{u}}{\partial \tau}\right)^{k+1/2}, \tag{4.22}$$

$$p^{k+1} = p^{k+1/2} + \frac{1}{2}\Delta\tau\left(\frac{\partial p}{\partial \tau}\right)^{k+1/2}. \tag{4.23}$$

Further, starting from eq. (4.22), the equation corresponding to eq. (4.19) becomes,

$$\left(\frac{d\mathbf{u}}{dt}\right)^{n+1} = \frac{3\Delta\tau}{2\Delta t}\left(\frac{\partial \mathbf{u}}{\partial \tau}\right)^{k+1/2} + \frac{3\mathbf{u}^k - 4\mathbf{u}^n + \mathbf{u}^{n-1}}{2\Delta t}, \tag{4.24}$$

where, the material derivative in the first term on the right hand side is replaced with a partial pseudo-time derivative. The eq. (4.20) also changes appropriately to no longer include the pseudo-time advection term, resulting in

$$\begin{aligned}\left(\frac{\partial \mathbf{u}}{\partial \tau}\right)^{k+1/2} \approx \Bigg\{ &-\frac{1}{\rho}\mathrm{grad}(p) + \nu\nabla^2\mathbf{u} + \mathbf{f} \\ &- \frac{(3\mathbf{u}^k - 4\mathbf{u}^n + \mathbf{u}^{n-1})}{2\Delta t}\Bigg\}\left(\frac{2\Delta t}{2\Delta t + 3\Delta\tau}\right).\end{aligned} \tag{4.25}$$

We note that usually the velocity of a particle in pseudo-time, $\hat{\mathbf{u}}$, is very small and this makes the changes to the position even smaller. This means that it is computationally efficient to fix the position of the particles in pseudo-time. In addition, even if we were to move the particles, we do not need to recompute the neighbor information.

Once the pseudo-time iterations are completed, we update the particle positions using,

$$\mathbf{r}^{n+1} = \mathbf{r}^n + \frac{\Delta t}{2}\left(\mathbf{u}^n + \mathbf{u}^{n+1}\right). \tag{4.26}$$

4.2.3 Steady State Solutions

We can use the dual-time to seek a solution to steady state problems. To do this we set the partial derivative in time to zero and retain only the pseudo time derivative. Further, we do not move the particles at all. We start with the following form of the continuity equation,

$$\frac{\partial p}{\partial \tau} + \frac{\partial p}{\partial t} + \mathbf{u} \cdot \text{grad}(\mathbf{u}) = -\rho c_s^2 \text{div}(\mathbf{u}). \tag{4.27}$$

We set the partial derivative in time to zero and this results in using the following form of equation for the evolution of the pressure,

$$\frac{\partial p}{\partial \tau} = -\mathbf{u} \cdot \text{grad}(\mathbf{u}) - \rho c_s^2 \text{div}(\mathbf{u}). \tag{4.28}$$

We note that we have not used the EDAC scheme in this case. Similarly, the eq. (4.8) for momentum reduces to,

$$\frac{\partial \mathbf{u}}{\partial \tau} = -\mathbf{u} \cdot \text{grad}(\mathbf{u}) - \frac{1}{\rho}\text{grad}(p) + \nu \nabla^2 \mathbf{u} + \mathbf{f}. \tag{4.29}$$

Here, we have moved the convection term to the right side and removed any partial time derivatives. This can be easily solved purely in pseudo-time while keeping the particles fixed in space. Technically, we could replace τ with t. However, the dual-time offers a convenient perspective for seeking a steady state solution iteratively.

The above approach to obtain steady state solutions is simple and not tied to any particular SPH scheme. Any scheme that uses a density or pressure evolution equation that is dependent on the divergence of the velocity will work. The approach is numerically efficient as it does not require any re-computation of neighbors. While simulating the steady-state problem, we iterate until the changes in pseudo-time become small enough.

4.3 SPH Discretization

The basic scheme discussed in the previous section should work for any particular SPH discretization of the momentum and pressure equations. In the following, we use a WCSPH formulation for the SPH discretization. We keep density fixed as per the original problem. We consider two different cases one where we move the particles in pseudo-time and the other where the particles are fixed in pseudo-time.

4.3.1 Moving Particles in Pseudo-Time

When the particles move in pseudo-time, the pressure evolution is computed using eq. (4.5). This is discretized as,

$$\begin{aligned}\frac{dp_i}{d\tau} =& \hat{\mathbf{u}}_i \cdot \sum_{j\in N(i)} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij} \\ &+ \sum_{j\in N(i)} \frac{m_j \rho_i}{\rho_j} c_s^2 \, \mathbf{u}_{ij} \cdot \nabla W_{ij} \\ &+ \sum_{j\in N(i)} \frac{4m_j}{(\rho_i + \rho_j)\rho_j} \nu_e \frac{p_{ij}}{(\mathbf{r}_{ij}^2 + \mathbf{h}_{ij}^2)} \nabla W_{ij} \cdot \mathbf{x}_{ij},\end{aligned} \tag{4.30}$$

where i denotes the i'th particle, $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$, $p_{ij} = p_i - p_j$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, $\mathbf{r}_{ij} = |\mathbf{x}_{ij}|$, $W_{ij} = W(\mathbf{r}_{ij}, h)$, is the SPH smoothing kernel with h as the smoothing length of the kernel and ν_e is the artificial viscosity coefficient used for the EDAC scheme (Ramachandran et al. 2019) as shown in eq. (4.6). The kernel is compact so the summation is over all the nearest neighbor particles that influence the particle i, $N(i)$. We note that the second term in the right hand side of eq. (4.5) corresponds to a density diffusion term employed in the δ-SPH scheme(Antuono et al. 2010).

For the momentum equation where the particles move in pseudo-time given in eq. (4.9), we can write,

$$\begin{aligned}\frac{d\mathbf{u}_i}{d\tau} + \frac{d\mathbf{u}_i}{dt} =& \hat{\mathbf{u}}_\mathbf{i} \cdot \sum_{j\in N(i)} \left(\mathbf{u}_i \frac{m_i}{\rho_i} + \mathbf{u}_j \frac{m_j}{\rho_j}\right) \nabla W_{ij} \\ &- \sum_{j\in N(i)} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij}\right) \nabla W_{ij} \\ &+ \sum_{j\in N(i)} m_j \frac{4\nu \nabla W_{ij} \cdot \mathbf{r}_{ij}}{(\rho_i + \rho_j)(r_{ij}^2 + \eta h_{ij}^2)} \mathbf{u}_{ij} + \mathbf{g}_i,\end{aligned} \tag{4.31}$$

where Π_{ij} is the artificial viscosity term (Monaghan 2005) added to the momentum equation is given by,

$$\Pi_{ij} = \begin{cases} \frac{-\alpha h_{ij} \bar{c}_{ij} \phi_{ij}}{\bar{\rho}_{ij}} & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} < 0, \\ 0 & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} \geq 0, \end{cases} \tag{4.32}$$

$$\phi_{ij} = \frac{\mathbf{u}_{ij} \cdot \mathbf{r}_{ij}}{r_{ij}^2 + \epsilon h_{ij}^2}, \tag{4.33}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$, $h_{ij} = (h_i + h_j)/2$, $\bar{\rho}_{ij} = (\rho_i + \rho_j)/2$, $\bar{c}_{ij} = (c_i + c_j)/2$, $\eta = 0.01$, and α is the artificial viscosity parameter. Note that in the above equations the viscosity is modeled using the discretization proposed by Morris et al. (1997).

We now write out the final form of the rate of change of the velocity in pseudo-time for the case where the particles are moving which we get by substituting eq. (4.19) in the momentum eq. (4.9), and discretizing the equations using the SPH formulation to get,

$$\begin{aligned}\frac{d\mathbf{u}_i}{d\tau} = \Bigg\{ & \hat{\mathbf{u}}_\mathbf{i} \cdot \sum_{j \in N(i)} \left(\mathbf{u}_i \frac{m_i}{\rho_i} + \mathbf{u}_j \frac{m_j}{\rho_j} \right) \nabla W_{ij} \\ & - \sum_{j \in N(i)} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla W_{ij} \\ & + \sum_{j \in N(i)} m_j \frac{4\nu \nabla W_{ij} \cdot \mathbf{r}_{ij}}{(\rho_i + \rho_j)(r_{ij}^2 + \eta h_{ij}^2)} \mathbf{u}_{ij} + \mathbf{g}_i \\ & - \frac{(3\mathbf{u}_i^k - 4\mathbf{u}_i^n + \mathbf{u}_i^{n-1})}{2\Delta t} \Bigg\} \left(\frac{2\Delta t}{2\Delta t + 3\Delta\tau} \right).\end{aligned} \tag{4.34}$$

It is important to note here that while we have used standard WCSPH discretization, the DTSPH formulation would work just as well with any other SPH discretization that is based on a weakly-compressible formulation. While in principle this may be extended to Riemann-solver based SPH formulations (Ferrari et al. 2009), this is not explored in the present work.

We observe that in eq. (4.34), we require the velocity at the current time and the previous time. When starting the simulations, if we do not have an exact solution, we assume that $\mathbf{u}^{-1} = \mathbf{u}^0$.

4.3.2 Fixed Particles in Pseudo-Time

In the case the particles are fixed in pseudo-time. The right hand side of the eq. (4.3) is discretized as,

$$\frac{\partial p_i}{\partial \tau} = \sum_{j \in N(i)} \frac{m_j \rho_i}{\rho_j} c_s^2 \, \mathbf{u}_{ij} \cdot \nabla W_{ij}. \tag{4.35}$$

In the case of the EDAC scheme, the right hand side of the eq. (4.7) is discretized as,

$$\frac{\partial p_i}{\partial \tau} = \sum_{j \in N(i)} \frac{m_j \rho_i}{\rho_j} c_s^2 \, \mathbf{u}_{ij} \cdot \nabla W_{ij} + \sum_{j \in N(i)} \frac{4 m_j}{(\rho_i + \rho_j)\rho_j} \nu_e \frac{p_{ij}}{(\mathbf{r}_{ij}^2 + \mathbf{h}_{ij}^2)} \nabla W_{ij} \cdot \mathbf{x}_{ij}, \tag{4.36}$$

For the momentum equation given in eq. (4.8), we can write,

$$\begin{aligned}\frac{\partial \mathbf{u}_i}{\partial \tau} + \frac{d\mathbf{u}_i}{dt} = &\sum_{j\in N(i)} -m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij}\right)\nabla W_{ij} \\ &+ \sum_{j\in N(i)} m_j \frac{4\nu \nabla W_{ij}\cdot \mathbf{r}_{ij}}{(\rho_i+\rho_j)(r_{ij}^2+\eta h_{ij}^2)}\mathbf{u}_{ij} + \mathbf{g}_i.\end{aligned} \tag{4.37}$$

By substituting eq. (4.37) in eq. (4.24) or eq. (4.31) in eq. (4.19), we get an equation for either $\frac{\partial \mathbf{u}}{\partial \tau}$ or $\frac{d\mathbf{u}}{d\tau}$. We can then use this to integrate the set of equations eq. (4.12) to eq. (4.17) or eq. (4.21) to eq. (4.23). Note that we first update the velocity and position of the particle as per eq. (4.10) and eq. (4.11).

The above equations govern the velocity and pressure of the fluids in the simulation. In order to satisfy the boundary conditions when solids are present we use an implementation of the boundary conditions presented in (Adami et al. 2012) where the pressure and the ghost velocity of the solid walls are set. Furthermore, following the work of (Hughes et al. 2010), we ensure that the pressure is always positive on the solid walls to prevent particles from sticking to them in our free surface simulations. We do not impose any specific free-surface boundary conditions.

4.3.3 Steady State Solutions

For the case of the steady state simulations, we do not move the particles in time and use the original distribution of particles. This is reasonable as steady solutions are usually sought where the geometry and boundaries are fixed. As discussed earlier, this leads to a very efficient solution procedure. For the pressure evolution we simply use eq. (4.35). For the momentum equation we use the following discretization for eq. (4.29),

$$\begin{aligned}\frac{\partial \mathbf{u}_i}{\partial \tau} = &\sum_{j\in N(i)} -m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij}\right)\nabla W_{ij} + \sum_{j\in N(i)} m_j \frac{4\nu \nabla W_{ij}\cdot \mathbf{r}_{ij}}{(\rho_i+\rho_j)(r_{ij}^2+\eta h_{ij}^2)}\mathbf{u}_{ij} \\ &+ \mathbf{g}_i - \mathbf{u}_i \cdot \sum_j^N \left(\mathbf{u}_i \frac{m_i}{\rho_i} + \mathbf{u}_j \frac{m_j}{\rho_j}\right)\nabla W_h,\end{aligned} \tag{4.38}$$

By solving these until the pseudo-time derivatives are small, we can obtain steady state solutions. This can be implemented very efficiently. The neighbors can be computed once and never need to be updated. The time step restrictions though continue to be as per the original weakly-compressible scheme.

4.3.4 Stability and Convergence

It is important to choose the real and pseudo time-steps carefully. We choose Δt such that $\Delta t\ \mathbf{u}_{\max} = Ch$, and C is around 0.25. This is similar to the time-steps used in the ISPH schemes.

We choose $1 < \beta < 20$, recall that $\beta = 1/M = c_s/u_{\text{ref}}$. We choose $\Delta\tau$ as $\Delta t/\beta$. The choice of these parameters is due to the following observations:

- The real time step is limited by the amount of permitted motion of the particles in one time step.
- The pseudo-time step can be seen to be essentially similar to the original weakly-compressible scheme and is therefore limited by the speed of sound. The pressure waves travel at the speed of sound and therefore the pseudo time-steps should be limited to around $\Delta t/\beta$.

We use the following approach to decide when to stop iterating in pseudo-time. The user specifies a particular tolerance, ϵ, which determines the termination of the pseudo-time iterations. During every pseudo-time iteration we compute the mean rate of change in the pressure, let us call this quantity $\delta p/\delta\tau$. We also compute the mean value of $|\hat{\mathbf{u}}|$ for all particles. When checking for convergence we ensure the following,

$$\begin{aligned} \frac{|\delta p/\delta\tau|\Delta t}{\rho c_s^2} &< \epsilon, \\ \frac{|\hat{\mathbf{u}}|}{u_{\text{ref}}} &< \epsilon. \end{aligned} \tag{4.39}$$

Note that we multiply the rate of change of pressure by Δt in order to ensure that the change over several pseudo-iterations would be accounted for.

Due to the inaccuracies of the SPH approximations and particle disorder, it is likely that the divergence does not become less than the tolerance and that the derivatives do not reduce. In order to prevent needless iterations we keep track of the changes in each pseudo-time iteration and stop iterations if the peak-to-peak relative changes in the last 3 or 4 iterations is less than 5%. This ensures that if the pressure and velocity do not change with increasing iterations we stop the iterations. This works very well in practice. We also stop iterating if there are more than 1000 iterations. In practice for reasonable tolerance values (larger than 10^{-5}) we typically have far less than 50 iterations per real time step. Our default tolerance is $\epsilon = 10^{-3}$.

4.3.5 Complete Algorithm

We summarize the proposed scheme using pseudo-code for clarity. The case of particles moving in pseudo-time is shown in algorithm 2. The case of particles fixed in pseudo-time is shown in algorithm 3.

Algorithm 2 Moving particles in pseudo-time, see Section 4.3.1.

while $t < t_{\text{final}}$ **do**
  for all particles **do**
    compute $\left(\frac{dp_i}{d\tau}\right)^k$ using eq. (4.30)
    compute $\left(\frac{d\mathbf{u}_i}{d\tau}\right)^k$ using eq. (4.34)
  for all particles **do**
    predict velocity using eq. (4.10)
    predict position using eq. (4.11)
  while check convergence using eq. (4.39) **do**
    for all particles **do**
      update $\left(\frac{dp_i}{d\tau}\right)^{k+\frac{1}{2}}$ using eq. (4.30)
      update $\left(\frac{d\mathbf{u}_i}{d\tau}\right)^{k+\frac{1}{2}}$ using eq. (4.34)
    for all particles **do**
      update $\hat{\mathbf{u}}^{k+1}$ using eq. (4.17)
    for all particles **do**
      update to $\mathbf{r}^{k+1}$, $\mathbf{v}^{k+1}$, and $\mathbf{p}^{k+1}$ using eqs. (4.12) and (4.16)
  update positions to $\mathbf{r}^{n+1}$ using eq. (4.26)
  shift particles using eq. (2.26)
  correct the velocities and pressure using eq. (2.28)

Algorithm 3 Fixed particles in pseudo-time, see Section 4.3.2.

while $t < t_{\text{final}}$ **do**
  for all particles **do**
    compute $\left(\frac{\partial p_i}{\partial \tau}\right)^k$ using eq. (4.36)
    compute $\left(\frac{\partial \mathbf{u}_i}{\partial \tau}\right)^k$ using eq. (4.37)
  for all particles **do**
    predict velocity using eq. (4.10)
    predict position using eq. (4.11)
  while check convergence using eq. (4.39) **do**
    for all particles **do**
      compute $\left(\frac{\partial p_i}{\partial \tau}\right)^{k+\frac{1}{2}}$ using eq. (4.36)
      update $\left(\frac{\partial \mathbf{u}_i}{\partial \tau}\right)^{k+\frac{1}{2}}$ using eq. (4.37)
    for all particles **do**
      update to $\mathbf{v}^{k+1}$, and $\mathbf{p}^{k+1}$ using eqs. (4.21) and (4.23)
  update positions to $\mathbf{r}^{n+1}$ using eq. (4.26)
  shift particles using eq. (2.26)
  correct the velocities and pressure using eq. (2.28)

4.4 Results and Discussion

In this section we perform various numerical experiments using standard benchmark problems. We explore the following specific questions using the Taylor-Green vortex which has an exact solution:

- Is it worth moving the particles in pseudo-time or can we freeze the particles? This has significant performance implications.
- What possible values of β can be used?
- What suitable values of the tolerance ϵ can be chosen and what does this imply for accuracy?

Once these are explored, a suite of test problems are simulated with the DTSPH scheme and compared with other schemes like the standard WCSPH (Monaghan 1994), transport velocity formulation (TVF) (Adami et al. 2013), entropically damped artificial compressibility (EDAC) (Ramachandran et al. 2019), and the δ-SPH scheme (Antuono et al. 2010). Except where noted, the DTSPH scheme uses the EDAC formulation in all the simulations below i.e. the pressure evolution uses eq. (4.36).

The TVF, EDAC, δ-SPH, and WCSPH schemes are part of the PySPH (Ramachandran et al. 2021b) framework. All the results presented below are automated and the code for the benchmarks is available at `https://gitlab.com/pypr/dtsph`. The tools used to automate the results are described in detail in (Ramachandran 2018). This allows us to automatically reproduce every figure and table in this work.

All the simulations are performed on a four core Intel (R) Core (TM) i5 − 7400 CPU with a clock speed of 3.00GHz. The problems are executed on four cores using OpenMP. We use the DTSPH scheme along with the EDAC in all results below unless explicitly mentioned otherwise.

4.4.1 Taylor-Green Vortex

The Taylor-Green vortex is a classical problem which is periodic along both x and y axes and has an exact solution. This is a particularly challenging problem for SPH (Adami et al. 2013; Fatehi et al. 2019; Rouzbahani et al. 2017) since the particles move along the streamlines towards a stagnation point leading to particle disorder.

The exact solution for the Taylor-Green vortex is given by

$$u = -Ue^{bt}\cos(2\pi x)\sin(2\pi y), \tag{4.40}$$

$$v = \quad Ue^{bt}\sin(2\pi x)\cos(2\pi y), \tag{4.41}$$

$$p = -U^2e^{2bt}(\cos(4\pi x) + \cos(4\pi y))/4, \tag{4.42}$$

where U is chosen as $1m/s$, $b = -8\pi^2/Re$, $Re = UL/\nu$, and $L = 1m$.

The Reynolds number Re is set to 100 and various cases are tested to better understand the scheme. For all the simulations, the quintic spline kernel is used with $h/\Delta x = 1.0$, no artificial viscosity is used. The following cases are considered,

- a comparison of results when particles are either advected or frozen in pseudo-time;
- the effect of changing β;

- the effect of changing the convergence tolerance, ϵ;
- comparison of results with different schemes; and
- comparison of the results with different number of particles and with different Reynolds numbers.

The results are compared against the exact solution. Particle plots are also shown wherever necessary as the error plots do not always reveal any particle disorder, particle clumping, or voiding occurring in the flow. In addition, we compare the performance of the schemes where the difference is noticeable.

We first discuss a simple method of initializing the particles that we use to compare all the schemes in a consistent and fair manner.

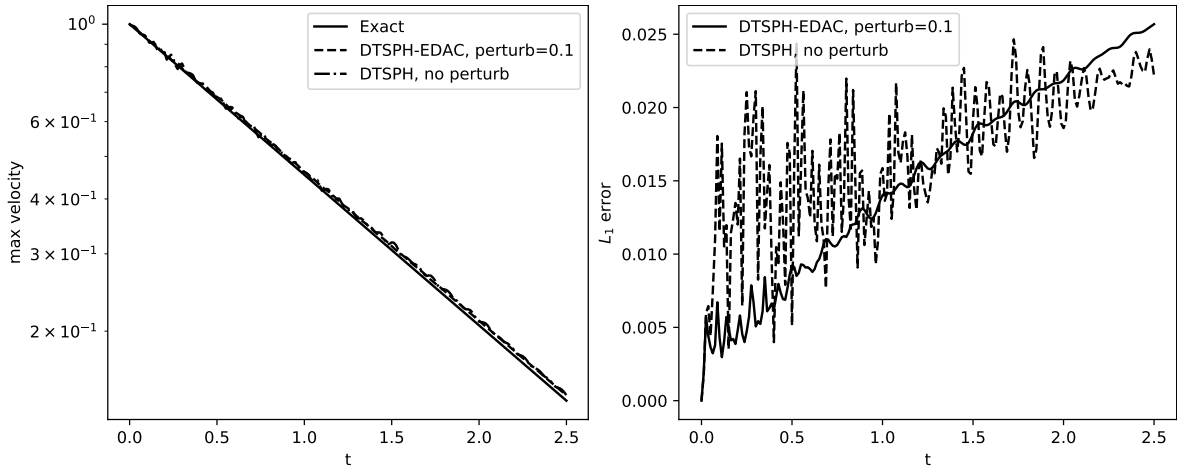
4.4.1.1 Perturbation in Initial Positions

As discussed earlier, it is not always easy to obtain good results for the Taylor-Green vortex. When the particles are initially distributed in a uniform manner, they tend to move towards the stagnation points and this often leads to severe particle disorder. In (Ramachandran et al. 2019), it was found that this problem can be reduced by introducing a small amount of noise in the initial particle distribution. To this end, a small random displacement is given to the particles with a maximum displacement of $\Delta x/10$. The random numbers are drawn from a uniform distribution and the random seed is kept fixed leading to the same distribution of particles for all cases with the same number of particles. This allows us to perform a fair comparison. Initially the particles are arranged in a 100×100 grid, with smoothing length of $h/\Delta x = 1$, $\Delta t = 0.00125$, $\epsilon = 10^{-4}$, and simulated for $2.5s$ with $\beta = 5$.

The decay rate is computed by computing the magnitude of maximum velocity $|\mathbf{u}_{\max}|$ at each time step, the L_1 error is computed as the average value of the difference between the exact velocity magnitude and the computed velocity magnitude, given as

$$L_1 = \frac{\sum_i |\mathbf{u}_{i,computed} - \mathbf{u}_{i,exact}|}{\sum_i |\mathbf{u}_{i,exact}|}, \tag{4.43}$$

where $\mathbf{u}_i$ is computed at the particle positions for each particle i in the flow. Figure 4.1a shows the decay of the velocity compared with the exact solution for the case where there is no initial perturbation and with a small amount of perturbation (of at most $\Delta x/10$). As can be clearly seen,



(a) Decay of maximum velocity with time.

(b) L_1 error of velocity magnitude with time.

Figure 4.1: Comparison of perturbation (of at most $\Delta x/10$) with and without any perturbation.

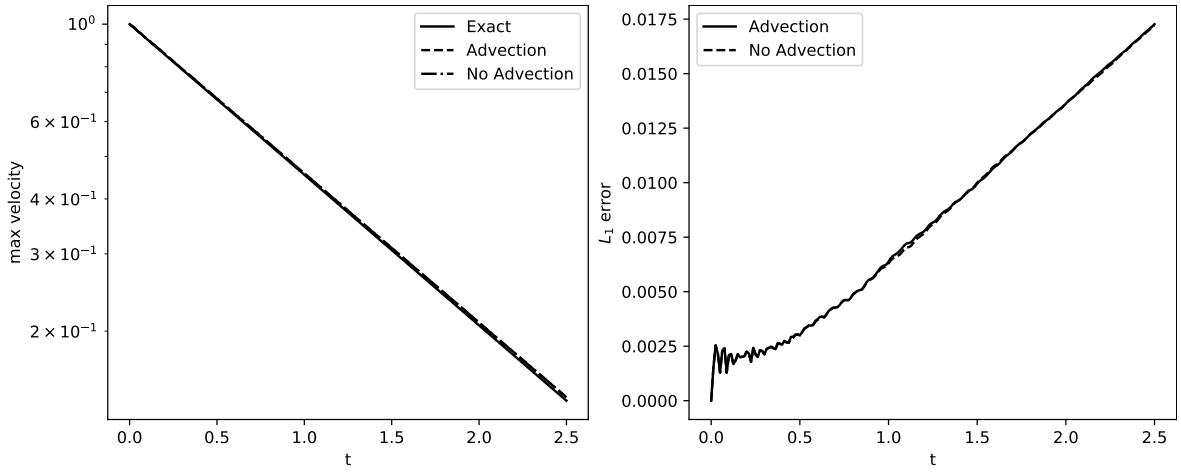
the introduction of the perturbation significantly improves the results. This is clearly seen in the L_1 norm of the error in the velocity magnitude in fig. 4.1b. While we have not shown this, the results are similar for simulations made using most other schemes including the new scheme, TVF, and the EDAC. Given this, we henceforth use a small initial perturbation for the results for the Taylor-Green vortex.

4.4.1.2 Advection of Particles in Pseudo-Time

As discussed earlier, it is important to study the effect of moving the particles in pseudo-time as against keeping them frozen in pseudo-time. If we find that there is no significant advantage gained by moving the particles in pseudo-time, we can simplify the implementation of the scheme as well as improve its performance considerably.

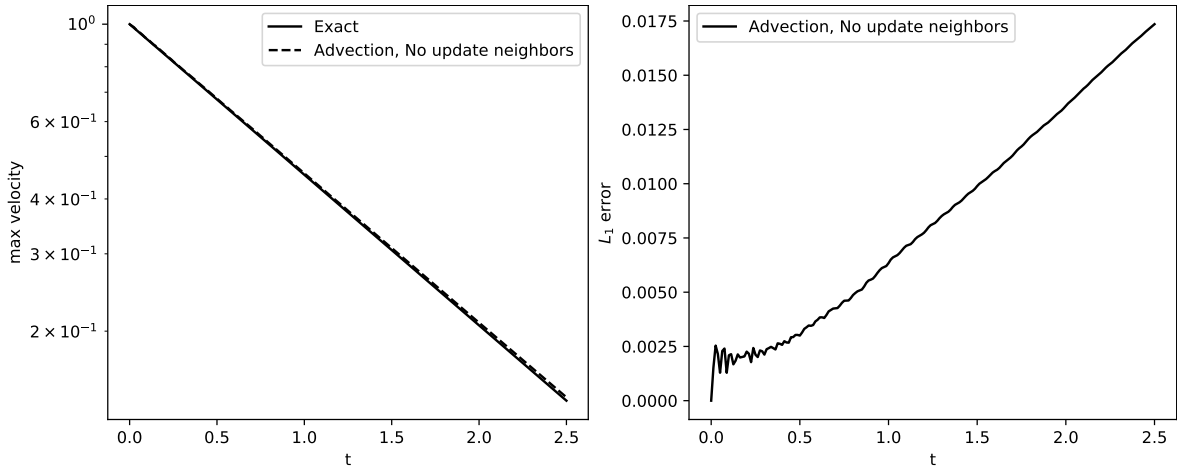
We consider the Taylor-Green vortex and compare the results of simulations where we advect the particles and where we hold them frozen. The rest of the parameters are held fixed. The same small perturbation is added to the initial particle position. Parameters used for this simulation are, initial particle spacing $\Delta x = \Delta y = 0.01$, Reynolds number $Re = 100$, the value of $\beta = 5$, time step $\Delta t = 0.00125$, and tolerance of $\epsilon = 10^{-6}$.

We recall that when we advect the particles in pseudo-time, we need to update the neighbors, however the displacements are very small and this is not necessary. We perform simulations to see if the differences are significant. Figure 4.2a shows the decay rate for the case with



(a) Decay of maximum velocity with time. (b) L_1 error in the velocity magnitude with time.

Figure 4.2: Velocity decay plot for $Re = 100$ for both advection and no advection of particles in pseudo time as compared with the exact solution for $Re = 100$.



(a) Decay of maximum velocity with time. (b) L_1 error in the velocity magnitude with time.

Figure 4.3: Comparison for decay rates with time and L_1 errors in velocity for advection and without advection cases while no update in the neighbor particles.

and without advection in pseudo-time while updating the neighbors. There are no noticeable differences in the results and the plots for each case lie on each other. This is also seen in fig. 4.2b which shows the L_1 error. Figures 4.3a and 4.3b show the decay rate and the L_1 error in the velocity magnitude while not updating the neighbors resulting in a similar conclusion that

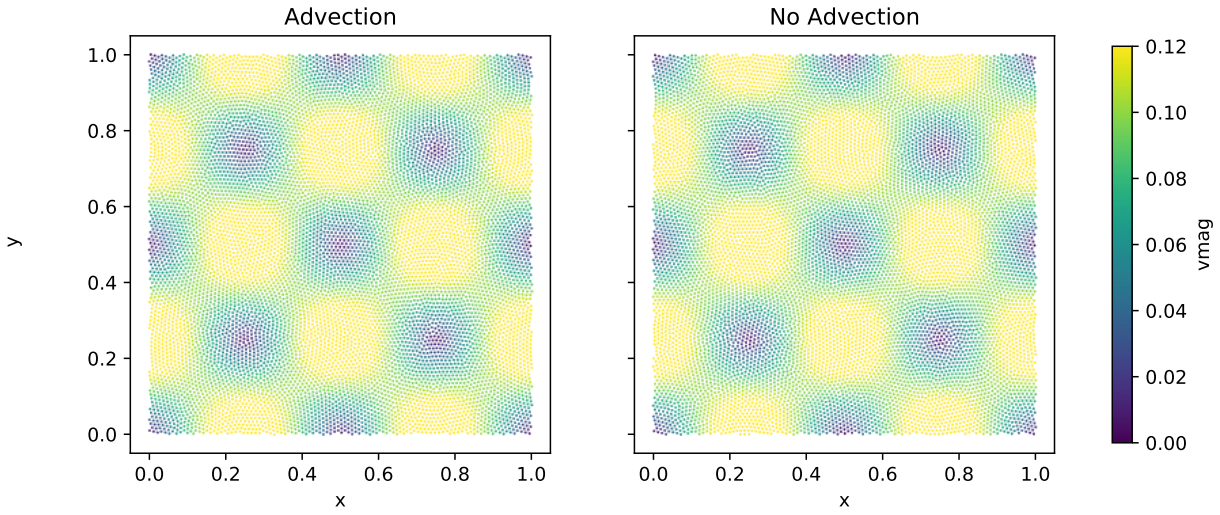


Figure 4.4: Particle plots for the Taylor-Green vortex with advection and without advection in pseudo time while updating the neighbors. Plots are shown at $t = 2.5s$ with $Re = 100$.

movement of particles in pseudo-time is too small to significantly influence the results.

Table 4.1: CPU time taken for 2.5 secs of Taylor-Green simulation with 100×100 particles, with advection and without advection in pseudo time.

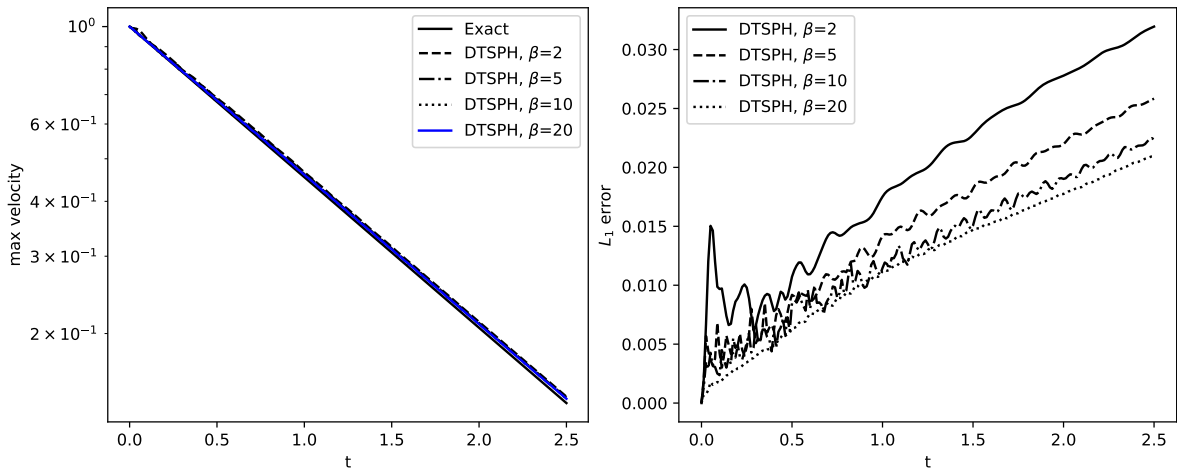
Scheme	CPU time (secs)
DTSPH frozen	326.87
DTSPH advect, no update neighbors	638.62
DTSPH advect, update neighbors	1934.21

While the accuracy is unaffected, the performance is significantly different as can be seen from table 4.1. This shows that advection of particles reduces performance by close to a factor of two. This increase in performance is largely due to the fact that we re-calculate the neighbor particles when we advect them. There is also some increase due to the additional computations required for the advection. Figure 4.4 shows the particle plots with color representing velocity magnitude for the case where the particles are advected and frozen. The results look identical. Based on these results, we do not advect the particles in pseudo-time for any of the other simulations.

4.4.1.3 Influence of β

The parameter β is the ratio of c_s/u_{ref}, as discussed earlier. The pseudo-time-step is also determined such that $\Delta t = \beta \Delta \tau$. In this section we consider the Taylor-Green vortex simulated at $Re = 100$ using 100×100 particles, using the new scheme with different values of β chosen between 2 and 20 for a tolerance $\epsilon = 10^{-4}$, and run for a simulation time of $t = 1$ sec.

Figures 4.5a and 4.5b show the decay rate and the L_1 error in the velocity for the different cases. From these it appears that β between 5 to 20 works well. We use a value of $\beta = 10$ in all simulations unless explicitly mentioned.



(a) Decay of maximum velocity with time.

(b) L_1 error of velocity magnitude with time.

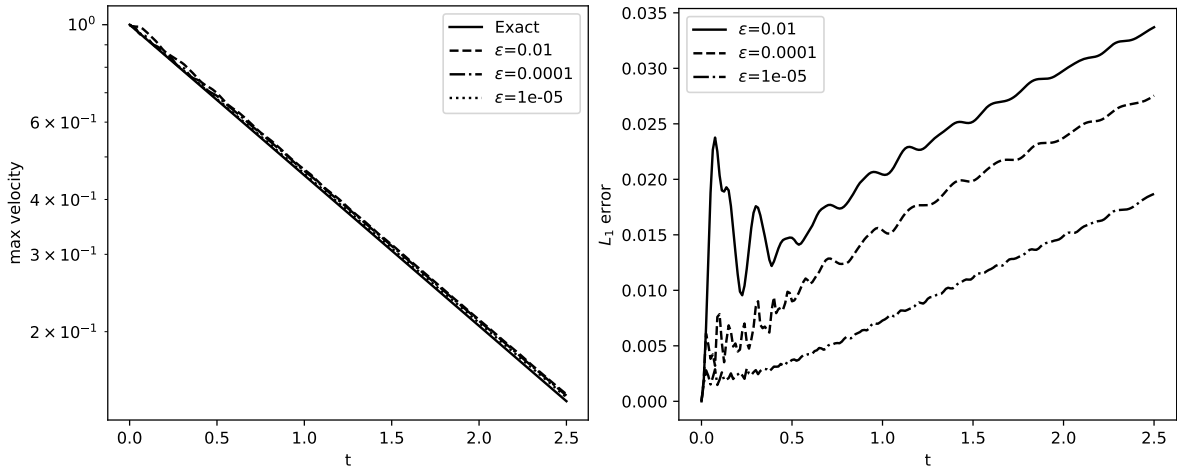
Figure 4.5: Decay rate of the maximum velocity and the L_1 error in the velocity magnitude for β values of $[2, 5, 10, 20]$ for the DTSPH scheme with an error tolerance of 10^{-4}.

4.4.1.4 Changing the Convergence Tolerance Parameter ϵ

We next choose $\beta = 10$ and vary the tolerance from 10^{-2} to 10^{-5}. Figure 4.6a shows the decay rates as the tolerance is changed and fig. 4.6b shows the L_1 error in the velocity. These results clearly show that reducing the tolerance improves the accuracy. However, we do see that the solutions are by-and-large robust to changes in ϵ over a very large range. As expected, increase in the tolerance leads to increase in the simulation time taken as seen in table 4.2.

Table 4.2: CPU time time taken for a simulation time of 2.5 secs with 100×100 particles with varying tolerance.

ϵ	CPU time (secs)
0.01	120.22
0.0001	133.87
1e-05	305.88



(a) Decay of maximum velocity with time. (b) L_1 error in the velocity magnitude with time.

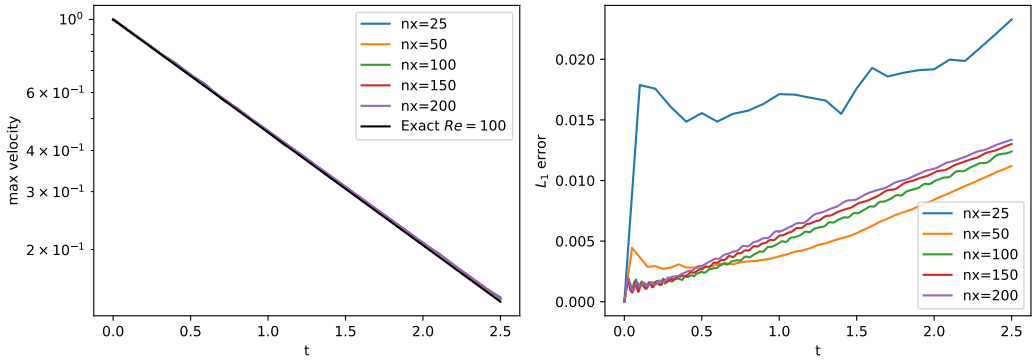
Figure 4.6: Comparison for various tolerance (ϵ) ranging from 10^{-2} to 10^{-5}.

4.4.1.5 Varying Reynolds Number

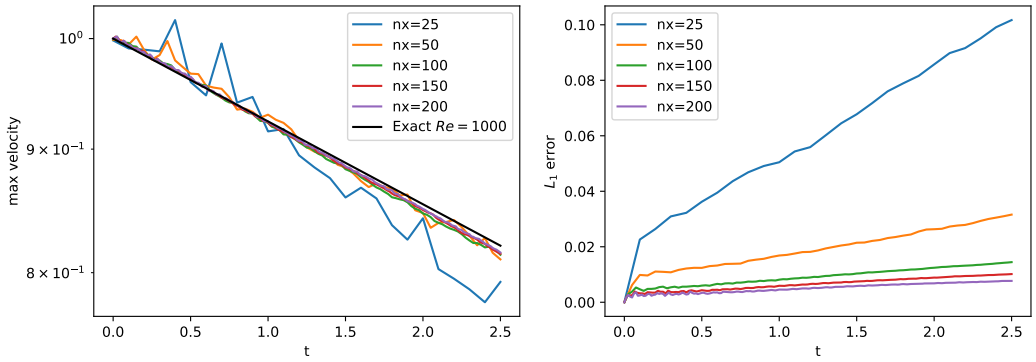
We simulate the problem at $Re = 1000$ using the most appropriate parameters based on the previous results. The DTSPH scheme is used with a quintic spline kernel, maximum initial random particle displacement of $\Delta x/10$, with a tolerance of $\epsilon = 10^{-3}$, and $\beta = 5$ for different initial particle arrangement of 25×25 to 200×200 and simulated for 2.5 secs.

Figure 4.7a shows the maximum velocity decay as well as the L_1 error of the velocity magnitude for the case of $Re = 100$. Figure 4.7b shows the same for $Re = 1000$. It can be seen that in the case of $Re = 1000$ that there is a clear reduction in the errors as the resolution is increased. In the case of $Re = 100$, it appears that the errors are lower for the 50×50 resolution and increase by a small amount as the resolution is increased. We believe that this occurs because of possible issues with the convergence of the discretization of the diffusive terms in

the governing equations. We point out that at $Re = 500$ we obtain similar convergence as in the case of $Re = 1000$ showing that issue is not with the convergence of the DTSPH scheme per se. These results show that the new scheme performs very well.



(a) Decay of maximum velocity with time and the L_1 error in the velocity magnitude for $Re = 100$.



(b) Decay of maximum velocity with time and the L_1 error in the velocity magnitude for $Re = 1000$.

Figure 4.7: Comparison of results for the Taylor-Green vortex using five different particle configuration between 25×25 to 200×200. Shown are the results for the Reynolds number of $Re = 100$ and 1000.

4.4.1.6 Comparison with other Schemes

Here we simulate the problem for $t = 2.5s$ for $Re = 100$ using the new DTSPH scheme comparing it with WCSPH, δ-SPH, and EDAC. The quintic spline kernel is used for all the

schemes with $h = \Delta x$. For all the cases the particles are perturbed by at most $\Delta x/10$. For DTSPH, we use $\beta = 10$ with a tolerance of $\epsilon = 10^{-4}$. We use an initial configuration of 100×100 particles.

As can be seen from the results shown in fig. 4.8, the new scheme is more accurate than the standard WCSPH scheme. The scheme is more accurate than the standard δ-SPH scheme. We note that we do not employ any form of shifting for the δ-SPH and WCSPH scheme cases. The scheme is not more accurate than the EDAC scheme as in the EDAC scheme, the particles are also regularized using the transport velocity formulation which significantly improves the results. As can be seen from the table 4.3, the new scheme is anywhere from 1.9 to 2.8 times faster than the other schemes.

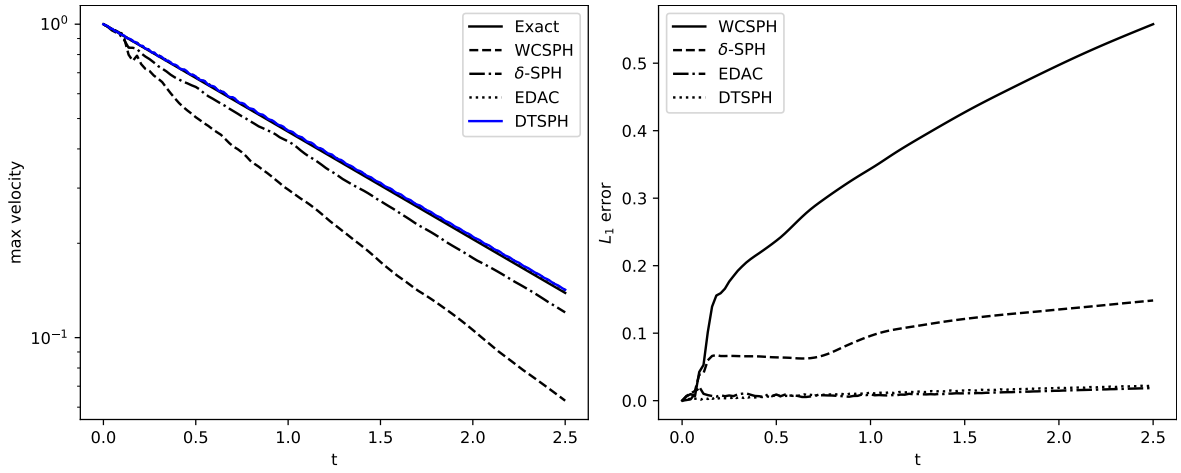
In fig. 4.9, we show the streamlines as well as the particle plot with the color indicating pressure at $t = 2.5$secs for the DTSPH case for $Re = 100$. As can be seen the particle distribution is smooth.

Table 4.3: CPU time time taken for a simulation time of 2.5 secs with 100×100 particles for various schemes.

Scheme	CPU time (secs)
DTSPH	124.49
WCSPH	239.70
EDAC	275.43
δ-SPH	347.98

4.4.2 Lid-Driven Cavity

We next consider the classic lid-driven cavity problem. This is a fairly challenging problem to simulate with SPH (Adami et al. 2013; Ghasemi V. et al. 2013; Lee et al. 2008). The fluid is placed in a unit square with a lid moving with a unit speed to the right. The bottom and side walls are treated as no-slip walls. The Reynolds number of the problem is given by $Re = \frac{u}{\nu}$, where u is the lid velocity in the x-direction. We use a quintic spline kernel with $h = \Delta x$. The problem is simulated at $Re = 100$ using a 50×50, 100×100, and 150×150 grid for a simulation time of $t = 10s$ until there is no change in the kinetic energy of the system. For the DTSPH scheme we use a $\beta = 10$ with a tolerance $\epsilon = 10^{-4}$. The results are compared with those of the TVF scheme (Adami et al. 2013) and the established results of Ghia et al. (1982). Figure 4.10, shows



(a) Decay of maximum velocity with time. (b) L_1 error in the velocity magnitude with time.

Figure 4.8: Comparison of DTSPH with other schemes for the simulation of Taylor-Green vortex, with $Re = 100$ and using 100×100 particles.

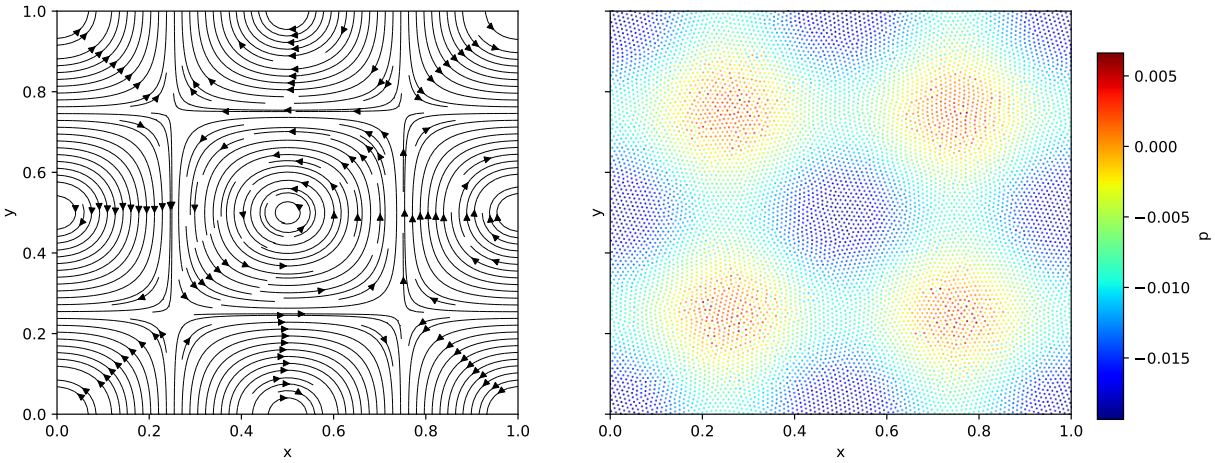


Figure 4.9: Particle plots for the Taylor-Green vortex, with 100×100 particles in the initial configuration, showing streamlines on the left and pressure on the left at $t = 2.5s$.

the centerline velocity profiles for u vs. y and v vs. x for different resolutions of particles. It is seen that the TVF scheme produces better results as expected. However, the results of the new scheme are in good agreement. In fig. 4.11, we show the particle distribution with the velocity magnitude on the left and streamlines on the right.

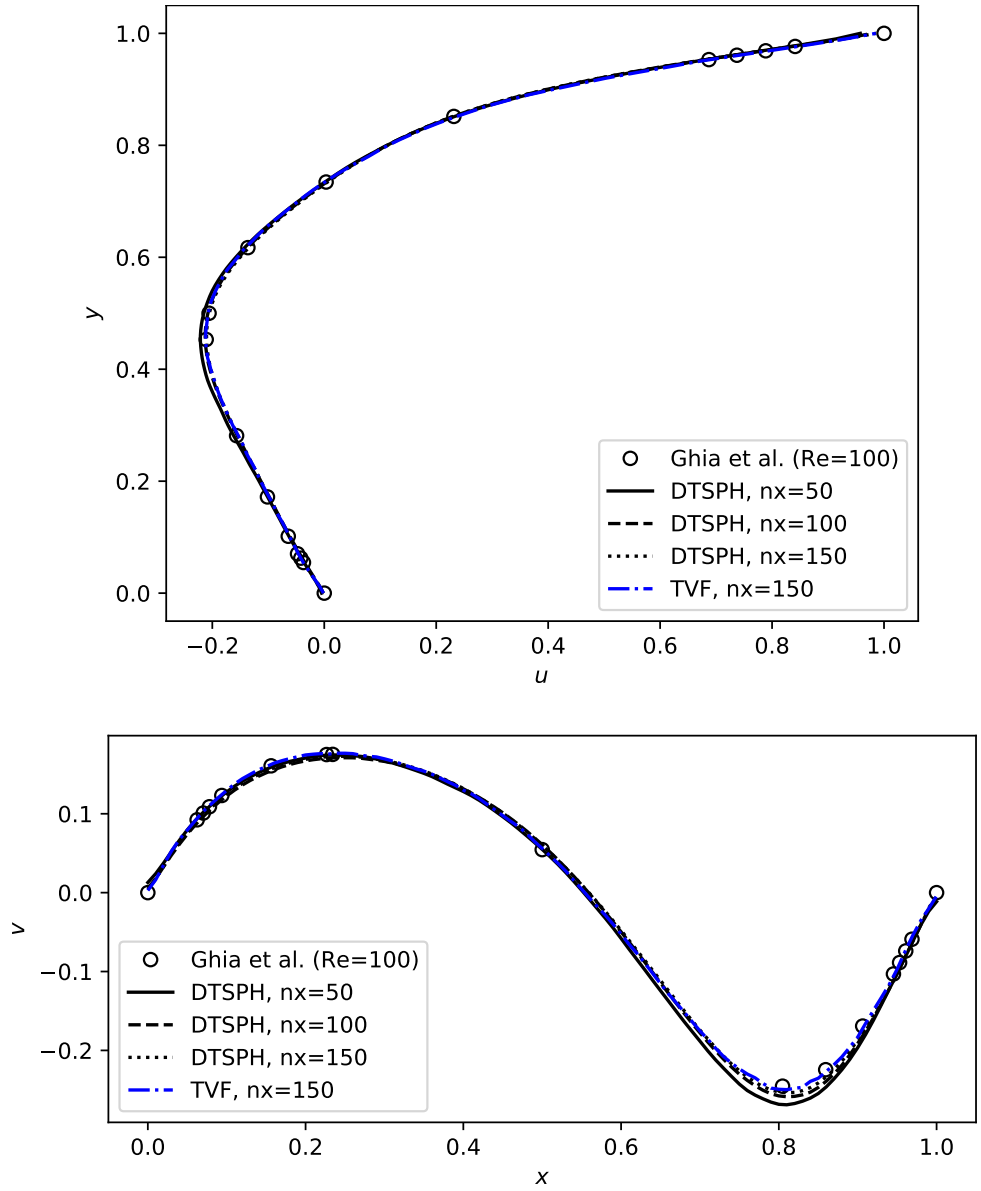


Figure 4.10: Velocity profiles u vs. y and v vs. x for the lid-driven cavity problem at $Re = 100$ with three initial particle arrangement of 50×50, 100×100, and 150×150. Here we compare DTSPH with TVF and the results of (Ghia et al. 1982).

4.4.2.1 Steady Lid-Driven cavity

In order to show that we are able to obtain steady state results, we employ the steady state equations discussed in Section 4.2.3 to solve the lid-driven cavity problem. We solve the problem

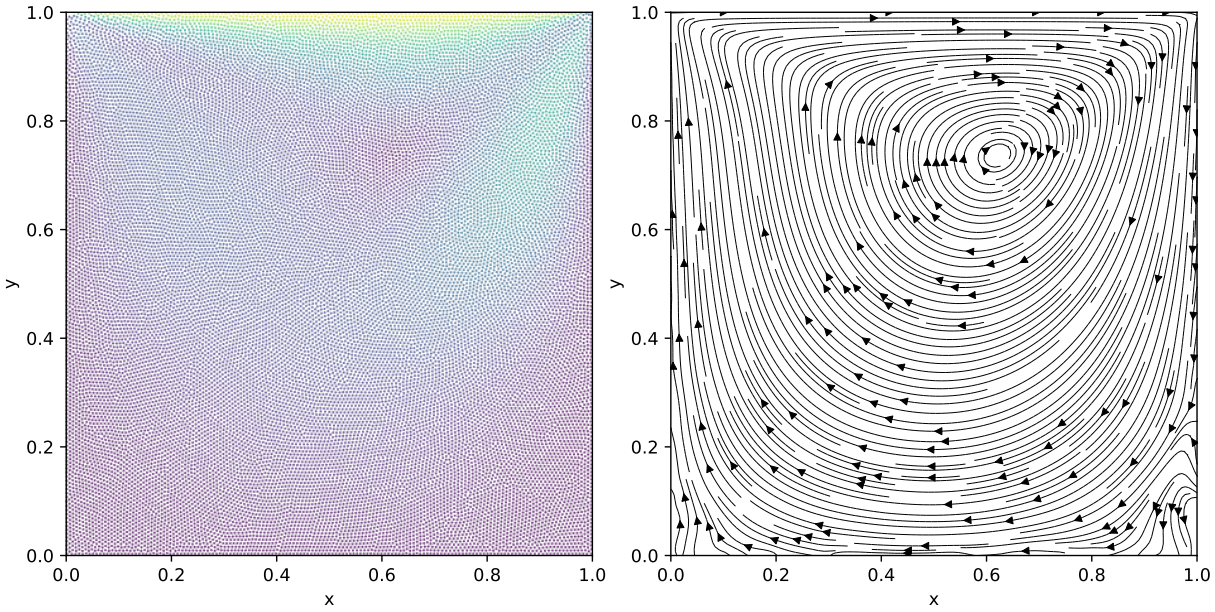


Figure 4.11: Particle plots for the cavity problem, with 150×150 particles in the initial configuration, showing the velocity magnitude on the left and streamlines on the right at $t = 10s$.

until there is no change in the kinetic energy of the system. We simulate the problem using a quintic spline kernel for $Re = 100$ and $Re = 1000$ using a 50×50, 100×100 and 150×150 particle grid. For $Re = 100$ we simulate the problem up to $\tau = 10$ and for $Re = 1000$ we simulate up to $\tau = 50$. Figure 4.12 shows the velocity profiles for the $Re = 100$ case and fig. 4.13 shows velocity profiles for the $Re = 1000$ case.

These results show that we are able to simulate internal flows very well using the new DTSPH scheme. We have also demonstrated that the steady-state equations also work very well. We next consider problems that involve a free-surface.

4.4.3 Square Patch

The square patch problem (Colagrossi 2005; Khayyer et al. 2013; Sun et al. 2017b) is a free surface problem where a square patch of fluid of side L is subjected to the following initial conditions,

$$\begin{aligned} u_0(x, y) &= \omega y, \\ v_0(x, y) &= -\omega x, \end{aligned} \tag{4.44}$$

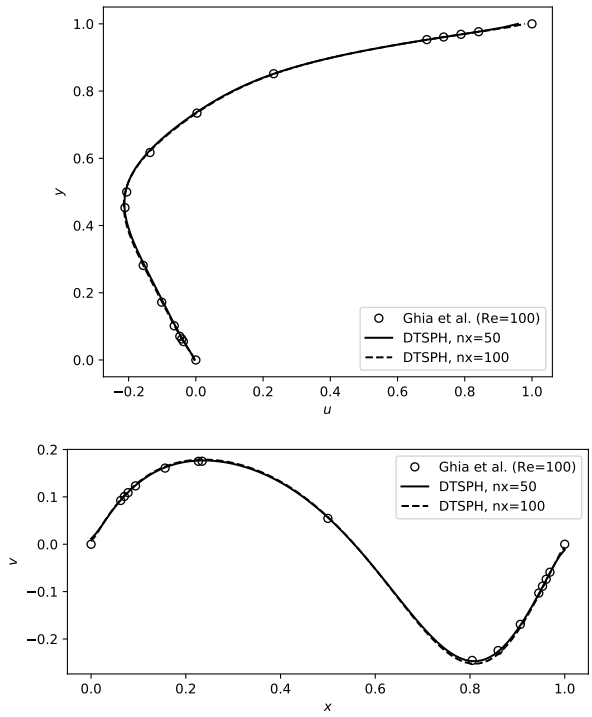


Figure 4.12: Velocity profiles for the lid-driven cavity using the steady state simulation procedure for $Re = 100$ with initial partial arrangement of 50×50, 100×100, and 150×150 compared with the results of Ghia et al. (1982).

$$p_0(x,y) = \rho \sum_m^\infty \sum_n^\infty -\frac{32\omega^2/(mn\pi^2)}{\left[\left(\frac{n\pi}{L}\right)^2 + \left(\frac{m\pi^2}{L}\right)^2\right]} \sin\left(\frac{m\pi x^*}{L}\right) \sin\left(\frac{n\pi y^*}{L}\right) m, n \in \mathbb{N}_{odd}, \tag{4.45}$$

where $X^* = x + L/2$ and $y^* = y + L/2$.

We simulate this problem for $t = 3s$ using the DTSPH, and EDAC schemes for comparison. In this case, the EDAC simulations also employ particle shifting (Lind et al. 2012). The quintic spline kernel with $h/\Delta x = 1.3$ is used for all the schemes, artificial viscosity $\alpha = 0.1$ is used for all the schemes. For the DTSPH scheme, $\beta = 10$ with a tolerance $\epsilon = 10^{-3}$ is used. Two different initial configurations of 50×50 and 200×200 particles are used. The particle distribution for each scheme at the end of $t = 3s$ is shown in fig. 4.14. The plots of DTSPH and EDAC are in good agreement with each other showing that the new scheme is as good as the EDAC scheme.

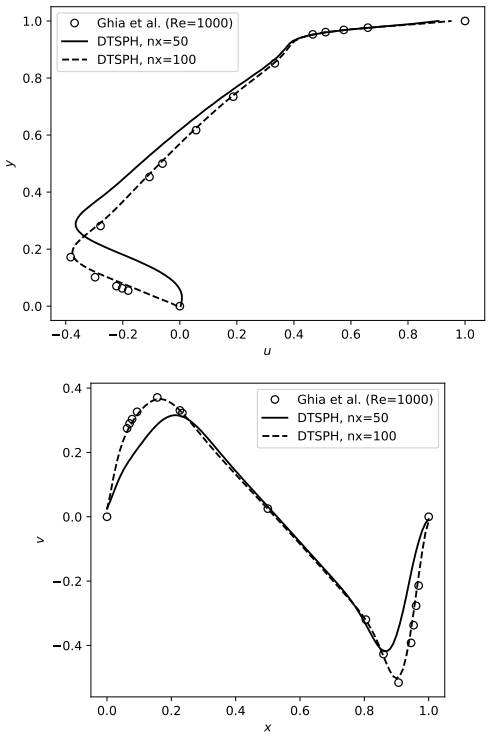


Figure 4.13: Velocity profiles for the lid-driven cavity using the steady state simulation procedure for $Re = 1000$ with initial partial arrangement of 50×50, 100×100, and 150×150 compared with the results of (Ghia et al. 1982).

4.4.4 Elliptical Drop

The elliptical drop problem was first solved in the context of the SPH by Monaghan (Monaghan 1994). This problem is also solved in the context of truly incompressible SPH (Khayyer et al. 2013; Lind et al. 2016; Rezavand et al. 2018). In this problem an initially circular drop of inviscid fluid having unit radius is subjected to the initial velocity field given by $-100x\mathbf{i} + 100y\mathbf{j}$. The outer surface is treated as a free surface. Due to the incompressibility constraint on the fluid there is an evolution equation for the semi-major axis of the ellipse.

This problem is simulated using the DTSPH, δ-SPH (without shifting), and EDAC (with shifting) respectively. An artificial viscosity parameter of $\alpha = 0.15$ is used for all the schemes. An error tolerance of $\epsilon = 10^{-4}$ is used for the DTSPH and scheme. $\beta = 10$, $\Delta x = 0.02$, $h = 1.3\Delta x$ and a quintic spline kernel is used for all the schemes. The simulation is run for $t = 0.0076s$.

Figure 4.15 shows the distribution of particles for different schemes. The colors indicate the

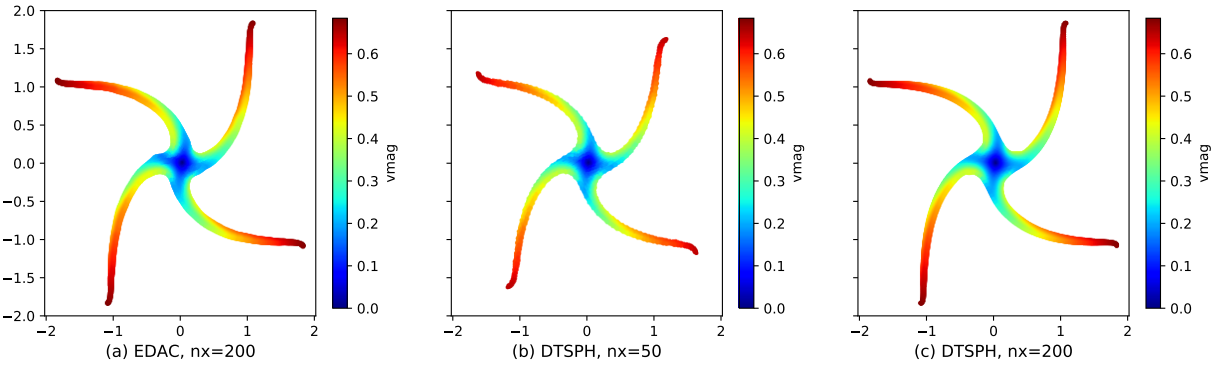


Figure 4.14: Particle distribution plots at $t = 3$ secs for the square patch problem. Artificial viscosity is used in all the schemes. Top row corresponds to 50×50 particles, and the bottom row corresponds to 100×100 particles. In column (a) EDAC scheme is used, with 200×200 particles, column (b) indicates DTSPH scheme with a tolerance of $\epsilon = 10^{-3}$, and 50×50 particles, and in column (c) DTSPH scheme is used with a tolerance of $\epsilon = 10^{-3}$, and 200×200 particles.

pressure. As can be seen, the DTSPH and EDAC results are similar. It is important to note that all the pressure values are in a similar range with none of the schemes exhibiting severe noise in the pressure. Figure 4.16, shows the evolution of the kinetic energy, the results are similar for

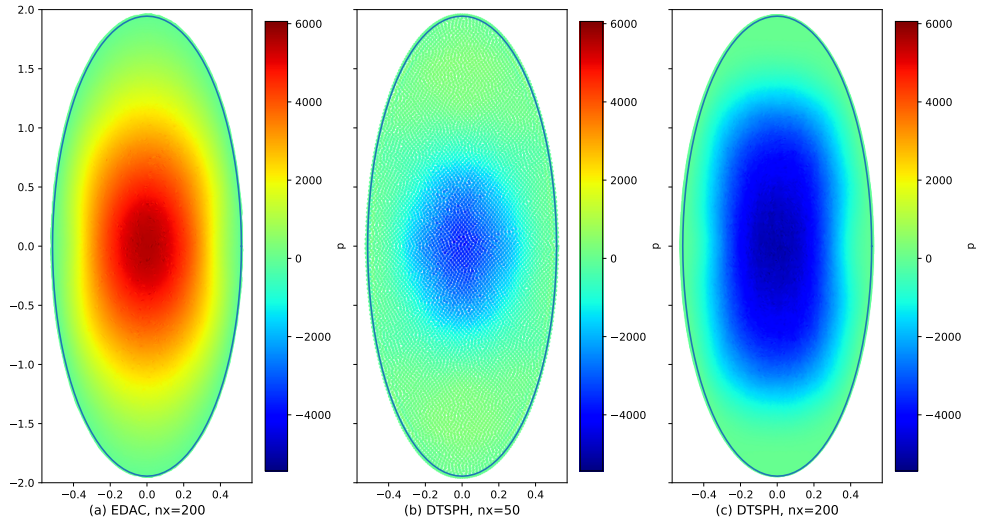


Figure 4.15: The distribution of particles for the elliptical drop problem at $t = 0.0076$ seconds. The plot (a) is with the EDAC using 200×200 particles. Plot (b) is that of the new DTSPH scheme with 50×50 particles, (c) uses DTSPH with 200×200. The solid blue line is the exact solution for the shape of the drop and the colors indicate the pressure.

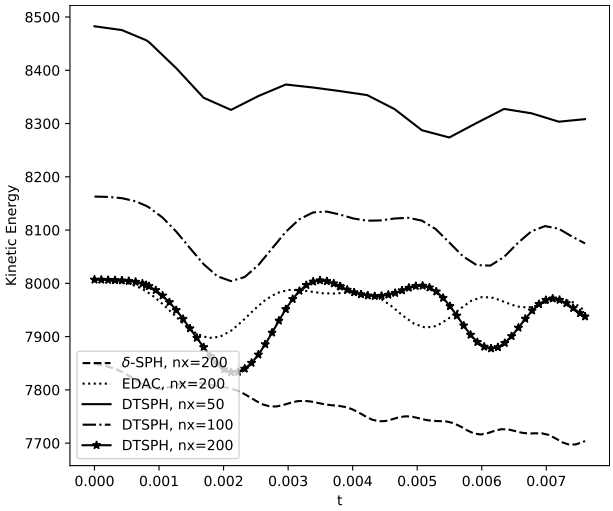


Figure 4.16: The kinetic energy with time of the Elliptical drop problem as computed with DTSPH, δ-SPH, and EDAC schemes.

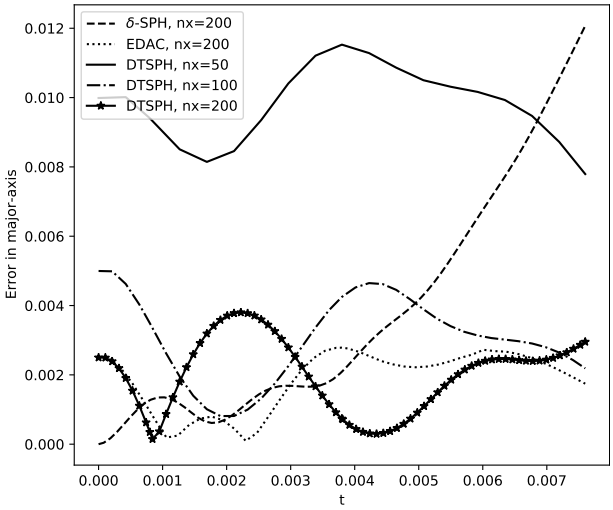


Figure 4.17: Error in computed size of semi-major axis of the elliptical drop problem compared with the exact solution for the DTSPH, δ-SPH and EDAC schemes.

DTSPH and EDAC schemes. This is to be expected. However it is interesting to note that the kinetic energy of the δ-SPH scheme decays faster than the other schemes. Figure 4.17 shows the error in the semi-major axis as compared to the exact solution. The DTSPH and EDAC both perform slightly better than the δ-SPH scheme. The results indicate that the new scheme performs well in comparison with state of the art weakly-compressible schemes.

4.4.5 Dam-break in Two Dimensions

A two-dimensional dam-break over a dry bed (Lee et al. 2008; Lind et al. 2016; Marrone et al. 2011) is considered next. The DTSPH and the standard EDAC schemes are compared. The simulation is performed for $1s$. The quintic spline kernel is used with $h/\Delta x = 1.0$, and an artificial viscosity of $\alpha = 0.1$ is used for all the schemes. A tolerance of $\epsilon = 10^{-4}$ is used for DTSPH.

The problem considered is described in (Lee et al. 2010) with a block of fluid column of height $h = 2m$, width $w = 1m$. The block is released under gravity which is assumed to be $-9.81m/s^2$.

For the DTSPH and EDAC schemes, the particle distribution is shown in fig. 4.18 at various times with color indicating pressure. Figure 4.19 shows the particle distribution with color indicating velocity magnitude at various times. The results of the new scheme seem largely comparable with that of the EDAC scheme. The results also show the improvements obtained by the addition of diffusive term in the pressure evolution equation. This significantly reduces the noise. The standard EDAC results are very similar to those of the δ-SPH and are hence not shown. Figure 4.20 plots the position of the toe of the dam versus time as compared with the results of the Moving Point Semi-implicit scheme of (Koshizuka et al. 1996). The results of the DTSPH scheme are in good agreement with those of the EDAC scheme.

4.4.6 Dam-break in Three Dimensions

A three dimensional case is shown to demonstrate the performance of the new scheme as compared to the EDAC scheme. This is an important case as the previous problems only require a smaller number of particles. We consider a three-dimensional dam-break over a dry bed with an obstacle. A cubic spline kernel is used for both the new scheme and the EDAC with $h/\Delta x = 1.3$ and artificial viscosity $\alpha = 0.1$. The problem is simulated for a total time of 1 second. We do not use any particle shifting in this case.

The problem considered is described in (Lee et al. 2010) with a block of fluid column of height $h = 0.55m$, width $w = 1.0m$ and length $l = 1.228m$. The container is $3.22m$ long. The block is released under gravity with an acceleration of $-9.81m/s^2$. Both schemes are simulated with a fixed time step. For both schemes we use a CFL of 0.25. The speed of sound for the EDAC case is set to $10\sqrt{2gh}$, where h is the height of the water column. For the DTSPH, we

use a time step of $\frac{0.25h}{\sqrt{2gh}}$, and use $\beta = 10, \epsilon = 10^{-3}$. The particle spacing, $\Delta x = 0.02$, leading to around 240, 000 particles in the simulation.

Figure 4.21 shows the particle distribution at various times for both the schemes. This indicate that the new scheme produces good results. Table 4.4 shows the time taken for the different 3D dam-break simulations. Depending on the tolerance chosen, we are able to obtain between a 1.7 to 7.16 fold improvement in performance as compared to the EDAC scheme. We note that as the tolerance ϵ is reduced, the DTSPH requires more iterations in pseudo-time in order to attain convergence and this reduces the performance. However even with lower tolerance values we are able to obtain very good results. In the next section we demonstrate the performance achievable with the new scheme for different problems.

Table 4.4: CPU time taken for different simulations of the 3D dam-break problem.

Scheme	ϵ	CPU time (secs)
DTSPH	0.001	563.19
DTSPH	0.0001	2404.97
EDAC	NA	4035.25

4.4.7 Performance

The performance of DTSPH is compared with that of other schemes for various problems. In table 4.5 we list the different problems and the speedup obtained by using the new scheme.

As we have seen before, for the dam-break problem in three dimensions it can be seen that DTSPH (with a tolerance of 10^{-3}) can be up to 7.16 times faster than the standard EDAC scheme. When a very low tolerance is used, the scheme is about 1.7 times faster this is only to be expected as lower tolerances require many more iterations to obtain a converged pressure. For the two-dimensional dam-break cases, the new scheme is about 3.5 times faster than the EDAC or the δ-SPH. For the unsteady cavity problem at $Re = 100$ with a 150×150 grid of particles, we get up to a 7 times performance improvement. This is quite significant since in these cases we have compared these cases where the effective Mach number is 0.1. Given this, the primary advantage with the DTSPH is that it can take a time step that is 10 times larger. Hence in this case a speed-up of close to 7 suggests that it is an efficient scheme.

Table 4.5: Speed-up obtained for different simulations when using the DTSPH scheme.

Problem	Scheme	Speed up
Dam-break 3D	EDAC vs. DTSPH ($\epsilon = 10^{-3}$)	7.16
Dam-break 3D	EDAC vs. DTSPH ($\epsilon = 10^{-4}$)	1.68
Dam-break 2D	EDAC vs. DTSPH	3.57
Dam-break 2D	EDAC vs. δ-SPH	3.66
Cavity	TVF vs DTSPH	6.87

The suite of benchmarks considered shows that the new scheme is robust, simulates a variety of problems, and is as accurate as the EDAC scheme. In addition it is very efficient and can be as much as seven times faster than the EDAC scheme in specific cases. Indeed, it is possible to improve the performance even more by caching the values of the kernel and kernel gradients during the pseudo-time iterations but we have not done this in the present work.

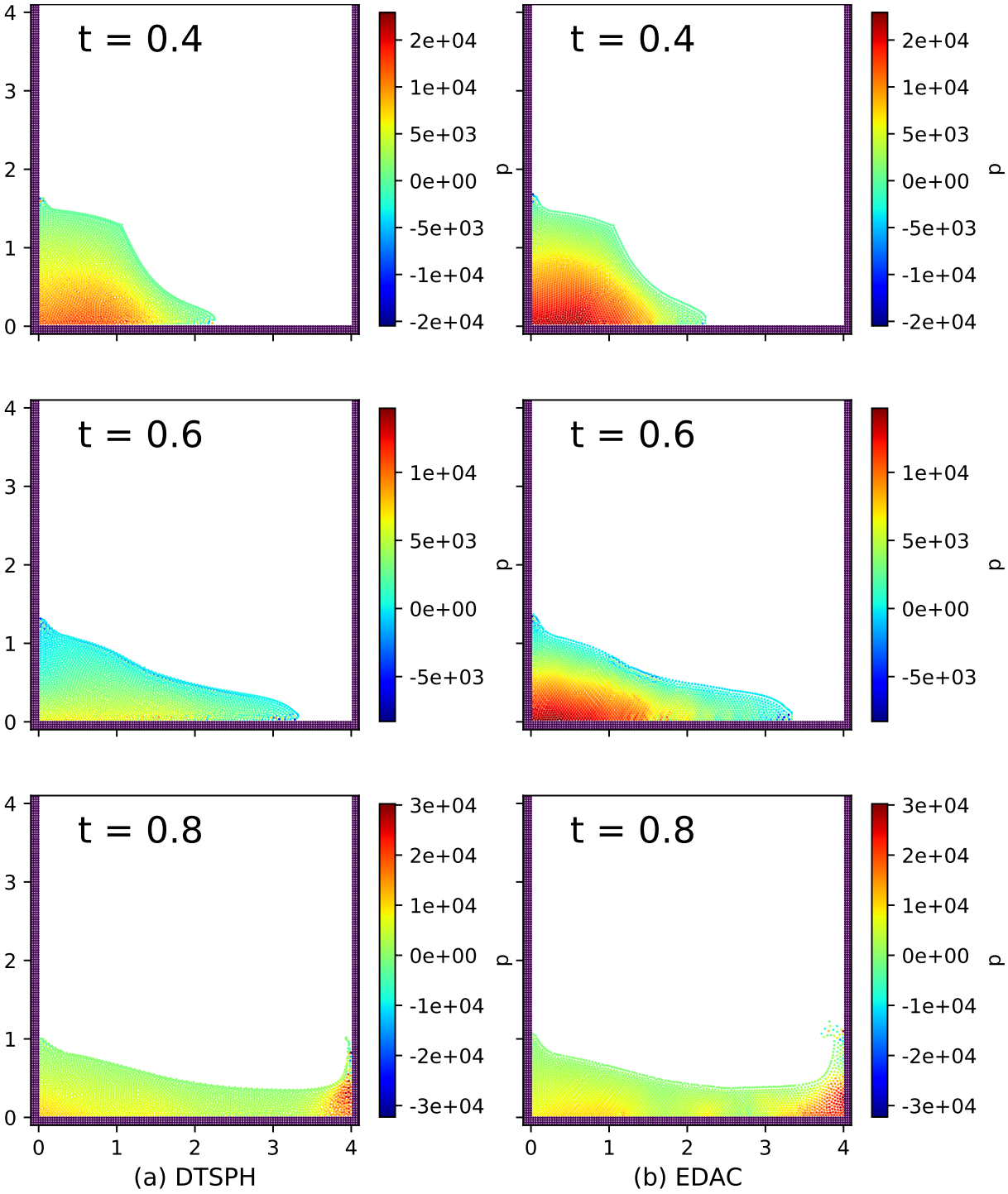


Figure 4.18: Particle distribution plots with color indicating pressure for the dam-break 2D problem at various times. DTSPH is shown on the left, and EDAC is shown on the right. Top row is at $t = 0.4$ secs, second row is at $t = 0.6$ secs, and bottom row is at $t = 0.8$ secs.

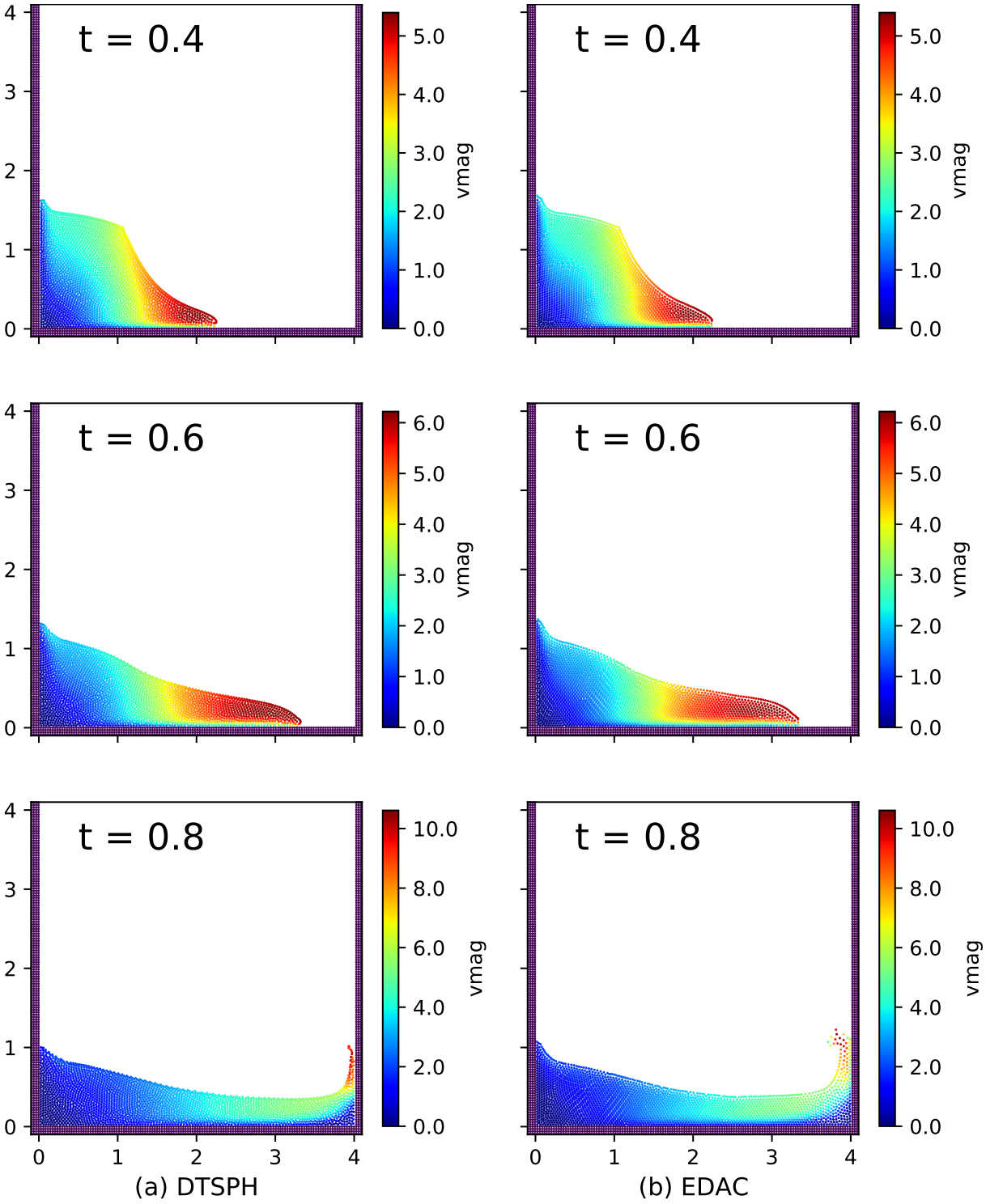


Figure 4.19: Particle distribution plots with color indicating velocity magnitude for the 2D dam-break problem at various times. DTSPH scheme is shown on the left, and EDAC is shown on the right. Top row is at $t = 0.4$ secs, the second row is at $t = 0.2$ secs, and bottom row is at $t = 0.8$ secs.

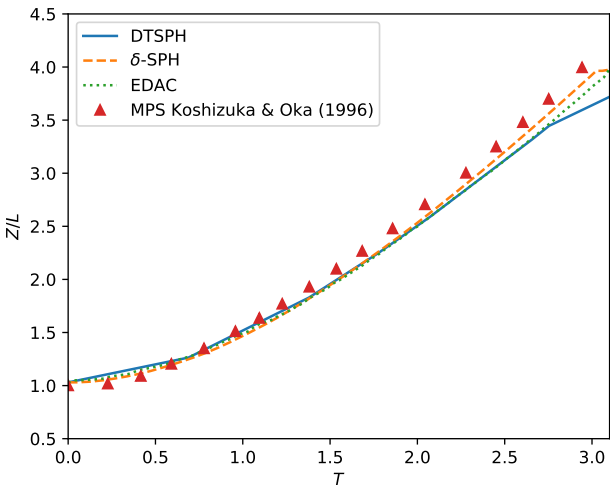
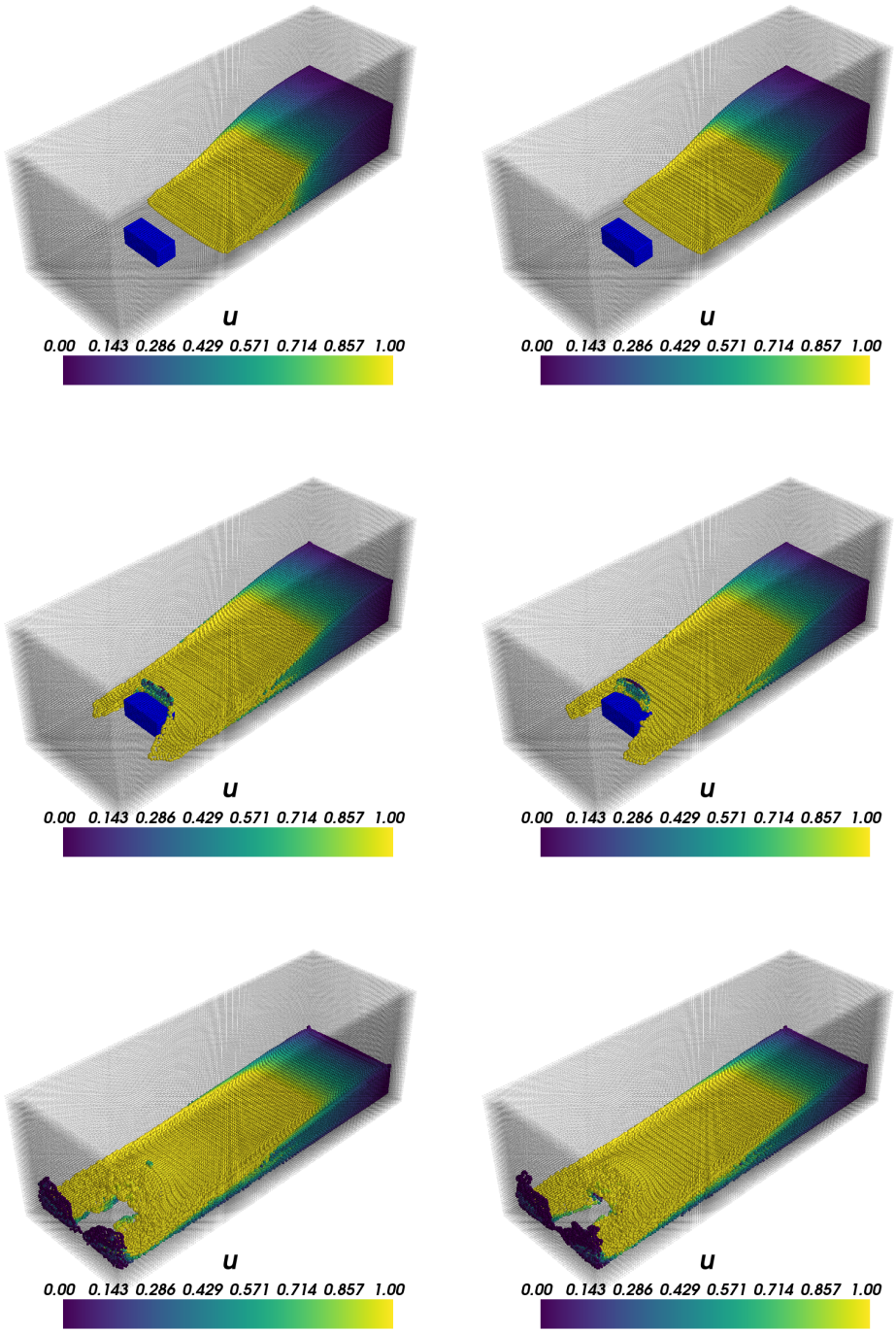


Figure 4.20: Position of the toe of the dam versus time of DTSPH, WCSPH, δ-SPH and EDAC, as compared with the simulation of (Koshizuka et al. 1996). Z is the distance of toe of the dam from the left wall and L is the initial width of the dam.



(a) EDAC scheme. (b) DTSPH scheme.

Figure 4.21: Dam-break 3D particle distribution at various times, first row is at $t = 0.4$ secs second row is at $t = 0.6$ secs and third row is at $t = 1.0$ secs. The left column is EDAC scheme and right column is the DTSPH scheme.

4.5 Summary

We propose a scheme called Dual-Time stepped SPH (DTSPH) that employs a dual-time stepping approach for incompressible fluid flow simulations. The method has been demonstrated with the EDAC formulation (Ramachandran et al. 2019). There are a few recent developments in the literature that employ a similar approach (Fatehi et al. 2019; Rouzbahani et al. 2017) but these implementations are not efficient despite the ability to use much larger time-steps than the WCSPH scheme. We find that by not moving the particles in pseudo-time we are able to improve performance significantly without loss of accuracy as attested by our simulations. We show that the scheme is robust and accurate. Through several benchmarks in two and three dimensions we show that the scheme produces results that are as accurate as the δ-SPH scheme (Antuono et al. 2010) as well as the EDAC scheme (Ramachandran et al. 2019) while being up to seven times faster. The method is matrix-free and may be implemented in the context of any explicit SPH scheme. The DTSPH method does introduce a few new parameters in the form of the term $\beta = \Delta t/\Delta\tau$ and the tolerance ϵ. We discuss how these parameters can be set based on rational considerations. For a reasonable choice of parameters, the performance of the method is comparable to that of incompressible SPH schemes. An open source implementation of the new scheme is provided and all the results in this chapter are fully reproducible.

5 Adaptive Resolution Smoothed Particle Hydrodynamics

Up your scale. Each pixel a million megabytes.

—— William Gibson, *Neuromancer*

5.1 Introduction

Adaptive resolution for SPH is still a challenging area of current research (Vacondio et al. 2020). Adaptive resolution allows us to capture the crucial features necessary at multiple scales. Adaptive resolution would appear to be naturally suited for Lagrangian meshless methods, but accuracy, computational efficiency, and robustness in handling a diverse range of problems, that are typically addressed by SPH, make it far from a straightforward solution.

The idea of splitting and merging particles is not new and has been successfully applied in the context of vortex methods (Rossi 1996). This technique has also been used by various researchers employing SPH for computer graphics (Adams et al. 2007; Desbrun et al. 1999; Solenthaler et al. 2011). However, the challenge in implementing this with the SPH method for incompressible and weakly-compressible fluids is to have a method that is both accurate and computationally efficient while minimizing the number of numerical parameters. This is a significant challenge. It bears emphasis that none of the existing adaptive resolution schemes for fluid flow problems with widely varying scales (Chiron et al. 2018; Sun et al. 2018; Vacondio et al. 2013b) feature an automatic adaptation strategy, nor do they inherently support complex

moving geometries or provide any ability to introduce solution-based adaptivity. The methods of (Hu et al. 2019; Spreng et al. 2020; Sun et al. 2021) do support moving geometries and solution-adaptivity however, they do not seem to have been applied to problems with widely-varying scales. Automatic adaptation is the automatic refinement of the fluid particles around a moving solid boundary without the user intervention.

In this work we propose a new approach which is automatically adaptive, computationally efficient, accurate, supports moving bodies, and solution adaptivity. The basic strategy is to split and merge particles carefully as originally proposed by Feldman et al. (2007) and Vacondio et al. (2013b). However, we adaptively merge particles to reduce the large particle counts. This is done in a computationally efficient manner, in parallel, and our simulations indicate that this approach is also accurate. We are thus able to control the particle refinement adaptively so as to effectively only double the number of particles in each refinement region while retaining accuracy. In addition, we carefully set the smoothing radius of the refined particles to be optimal for the particular refinement region thereby further improving performance in comparison to the approach of Vacondio et al. (2016). We use ideas inspired from the work of Yang et al. (2019) to automatically set the refinement criterion. This allows us to specify the geometry, a few parameters determining the maximum and minimum length scales and the algorithm automatically refines the particles as required. We discuss in some detail the algorithm proposed and show how it can be used to (i) handle complex geometries, (ii) specify user-specified refinement regions, (iii) handle moving geometries, and (iv) be used for solution-based adaptivity. We do not extensively explore solution-based adaptivity in this work but outline the basic ideas and demonstrate with a few simulations. The algorithms employed in this work are parallel and in principle may be executed on a General-Purpose Graphics Processing Unit (GPGPU).

We only consider two dimensional problems in this work but in principle the ideas naturally extend to three-dimensional cases. Although we use a modified EDAC-SPH (Ramachandran et al. 2019) scheme for the SPH discretization any similar method could be used. In the present work we do not consider any free-surface problems, however, our adaptive refinement algorithm can be easily extended to work with such problems. We consider several simple benchmark problems to demonstrate the accuracy of the approach. We then simulate the flow past a circular cylinder at a variety of Reynolds numbers in the range 40 - 9500 and compare these with some very well established simulations to show that the method is capable of resolving the necessary details with a minimum of particles. This translates to a proportional reduction in the computational time. The new method allows us to perform such computations with far fewer

particles than reported elsewhere with the SPH method. For the case of the flow past a circular cylinder the results we present require at least an order of magnitude fewer particles than those reported in (Sun et al. 2018) for a similar resolution. Finally, we note that none of the existing methods for adaptive SPH feature open source implementations. We provide a fully open source implementation based on the PySPH framework (Ramachandran et al. 2021b). The source code can be obtained from `https://gitlab.com/pypr/adaptive_sph`. Our work in this chapter is fully reproducible and every figure is automatically generated through the use of an automation framework (Ramachandran 2018).

5.2 Variable-h SPH Function Approximation

The following derivation[1] follows Vacondio et al. (2012a) (see also (Hernquist et al. 1989)). Let f be a scalar function on some domain $\Omega \subset \mathbb{R}^d$. Let h be the smoothing length function defined everywhere on Ω. In the gather formulation of function approximation, the smoothing length h is dependent on the point of approximation $\mathbf{r}$, whereas in the scatter formulation it would depend on the neighbors' position. Function approximation in the gather formulation is written as,

$$f(\mathbf{r}) = \int_\Omega f(\mathbf{s}) W(|\mathbf{r} - \mathbf{s}|, h(\mathbf{r}))\, \mathrm{d}\mathbf{s}. \tag{5.1}$$

Discretizing the domain and choosing an appropriate kernel gives the SPH approximation in the gather formulation,

$$f(\mathbf{r}_i) = \sum_j \frac{m_j}{\rho_j} f(\mathbf{r}_j) W(|\mathbf{r}_i - \mathbf{r}_j|, h(\mathbf{r}_i)). \tag{5.2}$$

where the summation is over all the neighbors of the particle at $\mathbf{r}_i$. Note that the smoothing length is a function of $\mathbf{r}_i$, the position of approximation.

If we choose density as the function in the above we get the summation density formula in the gather formulation,

$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h(\mathbf{r}_i)). \tag{5.3}$$

[1] The derivation given in Vacondio et al. (2012a) has typographical errors, but the final expression is correct.

If $\mathcal{V}$ is a vector field on Ω the directional derivative of the summation density along an arbitrary velocity vector $\mathbf{v} \in \mathcal{V}(\mathbf{r})$ is given by the formula,

$$\begin{aligned}\nabla\rho_i \cdot \mathbf{v} = D_{\mathbf{v}}\rho_i &= \sum_j m_j D_{\mathbf{v}} W(r_{ij}, h_i) \\ &= \sum_j m_j \left[\frac{\mathrm{d}W(r_{ij}, h_i)}{\mathrm{d}r_{ij}} D_{\mathbf{v}} r_{ij} + \frac{\mathrm{d}W(r_{ij}, h_i)}{\mathrm{d}h_i} D_{\mathbf{v}} h_i \right].\end{aligned} \tag{5.4}$$

To derive the expressions for the kernel derivatives with respect to r_{ij}, and h_i we need to note that the kernel function in SPH has a general form of,

$$W(r_{ij}, h_i) = \frac{1}{h_i^d} f(q_{ij}^i), \tag{5.5}$$

where, $q_{ij}^i = r_{ij}/h_i$. Then the expression for the derivative w.r.t. r_{ij} is,

$$\begin{aligned}\frac{\mathrm{d}W}{\mathrm{d}r_{ij}}(r_{ij}, h_i) &= \frac{1}{h_i^d} \frac{\mathrm{d}f}{\mathrm{d}q_{ij}}(q_{ij}^i) \frac{\mathrm{d}q_{ij}^i}{\mathrm{d}r_{ij}}, \\ &= \frac{1}{h_i^{d+1}} \frac{\mathrm{d}f}{\mathrm{d}q_{ij}}(q_{ij}^i), \\ &= \frac{1}{h_i} \frac{\mathrm{d}W_{ij}}{\mathrm{d}q_{ij}}(q_{ij}^i),\end{aligned} \tag{5.6}$$

where we used the formulas,

$$\frac{\mathrm{d}q_{ij}^i}{\mathrm{d}r_{ij}} = \frac{1}{h_i}, \text{ and } \quad \frac{\mathrm{d}W}{\mathrm{d}q}(q_{ij}^i) = \frac{1}{h_i^d} \frac{\mathrm{d}f}{\mathrm{d}q}(q_{ij}^i), \tag{5.7}$$

and the expression for the derivative w.r.t. h is,

$$\begin{aligned}\frac{\mathrm{d}W}{\mathrm{d}h_i}(r_{ij}, h_i) &= -\frac{d}{h^{d+1}} f(q_{ij}^i) + \frac{1}{h^d} \frac{\mathrm{d}f}{\mathrm{d}h_i}(q_{ij}^i), \\ &= -\frac{d}{h_i} W(r_{ij}, h_i) + \frac{1}{h^d} \frac{\mathrm{d}f}{\mathrm{d}q_{ij}}(q_{ij}^i) \frac{\mathrm{d}q}{\mathrm{d}h} \\ &= -\frac{d}{h} W(r_{ij}, h_i) - \frac{r_{ij}}{h^{d+2}} \frac{\mathrm{d}f}{\mathrm{d}q_{ij}}(q_{ij}^i) \\ &= -\frac{d}{h} W(r_{ij}, h_i) - \frac{r_{ij}}{h^2} \frac{\mathrm{d}W}{\mathrm{d}q_{ij}}(r_{ij}, h_i).\end{aligned} \tag{5.8}$$

Using eq. (5.6) in eq. (5.8) gives,

$$\frac{\mathrm{d}W}{\mathrm{d}h_i}(r_{ij}, h_i) = -\frac{d}{h_i} W(r_{ij}, h_i) - \frac{r_{ij}}{h_i} \frac{\mathrm{d}W}{\mathrm{d}r_{ij}}(r_{ij}, h_i). \tag{5.9}$$

We need expressions for $D_{\mathbf{v}} r_{ij}$ and $D_{\mathbf{v}} h_i$ to complete the final expression of eq. (5.4). $D_{\mathbf{v}} r_{ij}$ can be obtained by expanding the directional derivative,

$$D_{\mathbf{v}} r_{ij} = \frac{1}{r_{ij}} \mathbf{r}_{ij} \cdot \mathbf{v}. \tag{5.10}$$

To derive $D_{\mathbf{v}} h_i$, note that $h_i = (m_i/\rho_i)^{1/d}$, then taking the directional derivative will give,

$$\begin{aligned} D_{\mathbf{v}} h_i &= m_i^{1/d} \left(\frac{-1}{d}\right) \rho_i{}^{(-1/d-1)} D_{\mathbf{v}} \rho_i, \\ &= -\frac{h_i}{\rho_i d} D_{\mathbf{v}} \rho_i. \end{aligned} \tag{5.11}$$

Substituting eqs. (5.9) to (5.11) into eq. (5.4) gives,

$$D_{\mathbf{v}} \rho_i = \sum_j m_j \left[\frac{\mathrm{d}W(r_{ij}, h_i)}{\mathrm{d}r_{ij}} \frac{1}{r_{ij}} \mathbf{r}_{ij} \cdot (\mathbf{v}_i - \mathbf{v}_j) - \left(\frac{1}{\rho_i} W(r_{ij}, h_i) - \frac{r_{ij}}{\rho_i d} \frac{\mathrm{d}W}{\mathrm{d}r_{ij}}(r_{ij}, h_i) \right) D_{\mathbf{v}} \rho_i \right]. \tag{5.12}$$

Rearranging the terms in the equation gives,

$$\left(1 - \sum_j \frac{m_j}{\rho_i} W_{ij}\right) D_{\mathbf{v}} \rho_i = \sum_j m_j \left[\frac{\mathrm{d}W(r_{ij}, h_i)}{\mathrm{d}r_{ij}} \frac{1}{r_{ij}} \mathbf{r}_{ij} \cdot (\mathbf{v}_i - \mathbf{v}_j) + \frac{r_{ij}}{\rho_i\, d} \frac{\mathrm{d}W}{\mathrm{d}r_{ij}}(r_{ij}, h_i) D_{\mathbf{v}} \rho_i \right]. \tag{5.13}$$

Since, $\rho_i = \sum_j m_j W_{ij}$ rendering the left-hand side of the above equation to zero, thereby after re-arrangement,

$$D_{\mathbf{v}} \rho_i = \frac{1}{\beta_i} \left[\sum_j m_j \frac{\mathrm{d}W(r_{ij}, h_i)}{\mathrm{d}r_{ij}} \frac{1}{r_{ij}} \mathbf{r}_{ij} \cdot (\mathbf{v}_i - \mathbf{v}_j) \right] \tag{5.14}$$

where β_i[2] is defined as,

$$\beta_i = -\frac{1}{\rho_i\, d} \sum_j m_j r_{ij} \frac{\mathrm{d}W_{ij}}{\mathrm{d}r_{ij}}. \tag{5.15}$$

Noting that,

$$\nabla_i W(r_{ij}, h_i) = \frac{\mathrm{d}W}{\mathrm{d}r_{ij}}(r_{ij}, h_i) \frac{\mathbf{r}_{ij}}{r_{ij}}. \tag{5.16}$$

Substituting the above in the RHS of eq. (5.14) gives the final expression as,

$$D_{\mathbf{v}} \rho_i = \frac{1}{\beta_i} \left(\sum_j m_j \nabla_i W(x_{ij}, h_i) \cdot (\mathbf{v}_i - \mathbf{v}_j) \right). \tag{5.17}$$

[2]Note that the β_i defined here is the variable-h correction factor that is defined per particle, unlike the β of chapter 4 which is defined as c/u_{ref}, which is a global parameter defined once at the beginning of the simulation.

The definition of directional derivative is equivalent to the material-time derivative (Holzapfel 2002), we can write the LHS of the above equation as $\frac{\mathrm{D}\rho}{\mathrm{D}t}$, giving the continuity equation,

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \frac{1}{\beta_i}\left(\sum_j m_j \nabla_i W(x_{ij}, h_i)\cdot(\mathbf{v}_i - \mathbf{v}_j)\right). \tag{5.18}$$

In the next section we rewrite the governing equations in the variable-h SPH formulation.

5.3 The SPH Method

In this work we deal specifically with weakly-compressible flows. We use the Entropically Damped Artificial Compressibility (EDAC) method (Ramachandran et al. 2019) to simulate the weakly-compressible flows. The position update, pressure evolution, and momentum equations in the EDAC formulation are,

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \mathbf{u}, \tag{5.19}$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\rho c_s^2 \mathrm{div}(\mathbf{u}) + \nu_\mathrm{e}\nabla^2 p, \tag{5.20}$$

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f}, \tag{5.21}$$

where $\mathbf{r}$, $\mathbf{u}$, p, and t denotes the position, velocity, pressure, and time respectively. ρ is the density, ν is the kinematic viscosity of the fluid, c_s is the artificial speed of sound, $\mathbf{f}$ is the external body force, and ν_e is the EDAC viscosity parameter.

In order to further enhance the uniformity of the particles we use the transport velocity formulation (Adami et al. 2013), with the corrections incorporated (Adepu et al. 2021). Then the above equations are re-formulated as,

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \tilde{\mathbf{u}} \tag{5.22}$$

$$\frac{\tilde{\mathrm{d}}p}{\mathrm{d}t} = -\rho c_s^2 \mathrm{div}(\mathbf{u}) + \nu_\mathrm{e}\nabla^2 p + (\tilde{\mathbf{u}} - \mathbf{u})\cdot\nabla p \tag{5.23}$$

$$\frac{\tilde{\mathrm{d}}\mathbf{u}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} + \frac{1}{\rho}\nabla\cdot\rho(\mathbf{u}\otimes(\tilde{\mathbf{u}} - \mathbf{u})) + \mathbf{u}\,\mathrm{div}(\tilde{\mathbf{u}}) \tag{5.24}$$

where $\tilde{\mathbf{u}}$ refers to the transport velocity, and $\frac{\tilde{\mathrm{d}}(.)}{\mathrm{d}t} = \frac{\partial(.)}{\partial t} + \tilde{\mathbf{u}}\cdot\mathrm{grad}(.)$ is the material time derivative of a particle advecting with the transport velocity $\tilde{\mathbf{u}}$. The computation of the transport velocity is shown in Section 2.5.

Remark. In our numerical experiments with the Taylor-Green vortex we found that the addition of the divergence correction terms in the pressure evolution equation is crucial for accuracy. However, we find that the use of the last two terms in the momentum eq. (5.24) introduces noise where the particles are merged or split. Consequently, we do not use them in this work. We note that Sun et al. (2019a) observes that the effect of these terms in the momentum equation is minor.

We discretize the governing equations using variable-h SPH. The domain is discretized into particles whose spatial location is denoted by $\mathbf{r}_i$, where the subscript i denotes the index of an arbitrary particle. The mass of the particle, which vary as a function of space, is denoted by m_i, and its smoothing length by h_i. In the variable-h SPH the density is approximated by the summation density equation using a gather formulation (Hernquist et al. 1989; Vacondio et al. 2012b) written as,

$$\rho(\mathbf{r}_i) = \sum_j m_j W(|\mathbf{r}_i - \mathbf{r}_j|, h_i), \tag{5.25}$$

where, $W(|\mathbf{r}_i - \mathbf{r}_j|, h_i)$ is the kernel function.

The EDAC pressure evolution equation in variable-h SPH (see (Monaghan 2005; Vacondio et al. 2012b), for a derivation of the terms in the RHS) is given by,

$$\begin{aligned}\frac{\tilde{\mathrm{d}}p}{\mathrm{d}t}(\mathbf{r}_i) = \quad & \frac{\rho_0 c_s^2}{\beta_i}\sum_j \frac{m_j}{\rho_j}\mathbf{u}_{ij}\cdot\nabla W(r_{ij}, h_i) \\ & + \frac{1}{\beta_i}\sum_j \frac{m_j}{\rho_j}\nu_{\mathrm{e},ij}(p_i - p_j)(\mathbf{r}_{ij}\cdot\nabla W(r_{ij}, h_{ij})) \\ & + \sum_j m_j[(\tilde{\mathbf{u}}_i - \mathbf{u}_i)\cdot(P_i\nabla W(r_{ij}, h_i) + P_j\nabla W(r_{ij}, h_j))],\end{aligned} \tag{5.26}$$

where ρ_0 is the reference density, p_i is the pressure of particle i, ρ_j is the density of the j^{th} particle computed using summation density eq. (5.25), $\mathbf{u}_{ij} = (\mathbf{u}_i - \mathbf{u}_j)$, $r_{ij} = |\mathbf{r}_{ij}| = |\mathbf{r}_i - \mathbf{r}_j|$, β_i is the variable-h correction term (Vacondio et al. 2012b). P_i and P_j are given by,

$$P_i = \frac{(p_i - p_{\mathrm{avg},i})}{\rho_i^2\beta_i}, \quad P_j = \frac{(p_j - p_{\mathrm{avg},j})}{\rho_j^2\beta_j}, \tag{5.27}$$

here we employ the pressure reduction technique proposed by Basa et al. (2009), where, the average pressure is computed as,

$$p_{\mathrm{avg},i} = \frac{\sum_{j=1}^{N_i} p_j}{N_i}, \tag{5.28}$$

where N_i is the number of neighbors for a particle with index i, and

$$\nabla W(r_{ij}, h_{ij}) = \left(\frac{\nabla W(r_{ij}, h_i) + \nabla W(r_{ij}, h_j)}{2} \right). \tag{5.29}$$

The EDAC viscosity of the pressure diffusion term in the EDAC equation with the SPH discretization is given by,

$$\nu_{\mathrm{e},i} = \frac{\alpha_\mathrm{e} c_s h_i}{8}, \tag{5.30}$$

where $\alpha_\mathrm{e} = 1.5$ is used in all our simulations. Since this is a function of the smoothing length, which is varying in space, we use the approach of Cleary et al. (1999) to model the pressure diffusion term where,

$$\nu_{\mathrm{e},ij} = 4 \frac{\nu_{\mathrm{e},i} \nu_{\mathrm{e},j}}{(\nu_{\mathrm{e},i} + \nu_{\mathrm{e},j})}. \tag{5.31}$$

The momentum equation in the variable-h SPH discretization is given by,

$$\begin{aligned} \frac{\tilde{\mathrm{d}}\mathbf{u}}{\mathrm{d}t}(\mathbf{r}_i, t) = &- \sum_j m_j \left((P_i + A_i) \nabla W(r_{ij}, h_i) + (P_j + A_j) \nabla W(r_{ij}, h_j) \right) \\ &+ \frac{1}{\beta_i} \sum_j m_j \frac{4\nu}{(\rho_i + \rho_j)} \frac{\mathbf{r}_{ij} \cdot \nabla W(r_{ij}, h_{ij})}{(|\mathbf{r}_{ij}|^2 + \eta)} \mathbf{u}_{ij} \\ &- \frac{1}{\beta_i} \sum_j \frac{m_j}{\rho_j} [(\tilde{\mathbf{u}}_{ij} - \mathbf{u}_{ij}) \cdot \nabla W(r_{ij}, h_i)] \mathbf{u}_i \end{aligned} \tag{5.32}$$

where,

$$A_i = \frac{1}{\rho_i \beta_i} \mathbf{u}_i \otimes (\tilde{\mathbf{u}}_i - \mathbf{u}_i), \quad A_j = \frac{1}{\rho_j \beta_j} \mathbf{u}_j \otimes (\tilde{\mathbf{u}}_j - \mathbf{u}_j), \tag{5.33}$$

and $\eta = 0.001 h_i^2$ is a small number added to ensure a non-zero denominator in case when $i = j$.

Remark. We do not employ any artificial viscosity in our benchmark cases. We note that the proposed scheme is not conservative due to shifting, the variable-h correction terms, and the non-standard form of the pressure gradient.

5.3.1 Particle Shifting

We use a limited form (Oger et al. 2016a; Sun et al. 2017a) of the particle shifting technique of Lind et al. (2012) where the positions of the particles are shifted by,

$$\mathbf{r}_i' = \mathbf{r}_i + \theta \delta \mathbf{r}_i, \tag{5.34}$$

where, a parameter θ is used to limit the movement of the particles ($\theta = 1$ recovers the shifting presented in Section 2.5). We used ρ_0 in the volume of eq. (2.27) as we found that using ρ_j makes the shifting less effective. We limit the shifting by restricting the movement of particle which is shifted by more than 25% of its smoothing length:

$$\theta = \begin{cases} \frac{0.25h_i}{|\delta \mathbf{r}_i|} & \text{if } |\delta \mathbf{r}_i| > 0.25h_i, \\ 1 & \text{otherwise.} \end{cases} \tag{5.35}$$

We employ shifting while solving the fluid equations and also after our adaptive refinement procedure. Since we use the transport velocity scheme which already accounts for the shifting no additional correction is necessary. However, after the adaptive refinement procedure and subsequent shifting we correct the fluid properties by using a Taylor series approximation.

5.3.2 Boundary Conditions

We employ periodic, no-slip, free-slip, no-penetration and the inlet-outlet boundary conditions in our test cases. We enforce periodic boundary conditions by the use of ghost particles, onto which the properties are directly copied from the particles exiting the domain through a periodic boundary.

For the no-slip, free-slip and no-penetration boundary conditions we use the dummy particle technique of Adami et al. (2012). Dummy particles placed in uniform layers are used to discretize the wall. The no-penetration is implicitly enforced by using the wall velocity in the EDAC equation (Adami et al. 2012). For the no-slip or free-slip we extrapolate the values of velocity of the fluid onto the dummy wall particles by,

$$u_w = 2\mathbf{u}_i - \hat{\mathbf{u}}_i, \tag{5.36}$$

where the subscript w denotes the dummy wall particles, $\mathbf{u}_i$ is the prescribed wall velocity, and

$$\hat{\mathbf{u}}_i = \frac{\sum_j \mathbf{u}_j W(r_{ij}, h_{ij})}{\sum_j W(r_{ij}, h_{ij})} \tag{5.37}$$

is the Shepard extrapolated velocity of the fluid particles indexed by j onto the dummy wall particles i. The pressure on the wall is calculated from the fluid, to accurately impose the pressure gradient, by,

$$p_w = \frac{\sum_f p_f W(r_{wf}, h_{wf}) + (\mathbf{g} - \mathbf{a}_w) \cdot \sum_f \rho_f \mathbf{r}_{wf} W(r_{wf}, h_{wf})}{\sum_f W(r_{wf}, h_{wf})}, \tag{5.38}$$

where the subscript f denotes the fluid particles, $\mathbf{a}_w$ is the acceleration of the wall, $r_{wf} = |\mathbf{r}_w - \mathbf{r}_f|$, and $h_{wf} = (h_w + h_f)/2$.

For the inlet and outlet we use the non-reflecting boundary condition of Lastiwka et al. (2009). To begin with, compute the characteristic properties, referred to as J_1, J_2, and J_3 in aforementioned article, of the fluid particles. Then, we extrapolate the characteristic variables of the fluid onto the inlet and outlet particles using Shepard interpolation. Finally, we determine the fluid dynamical properties from the characteristic variables.

5.3.3 Force Computation

We compute the forces on the circular cylinder in the flow past a circular cylinder simulation and evaluate the coefficients of lift and drag. Specifically, we compute the forces due to the pressure and the skin-friction on the cylinder by evaluating,

$$\mathbf{f}^{\text{solid}} = m^{\text{solid}} \left(-\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla \mathbf{u} \right), \tag{5.39}$$

which in the variable-h SPH discretization is written as,

$$\begin{aligned} \mathbf{f}_i^{\text{solid}} = & \underbrace{-m_i^{\text{solid}} \sum_j m_j \left(P_i \nabla W(r_{ij}, h_i) + P_j \nabla W(r_{ij}, h_j) \right)}_{\mathbf{f}_{i,\text{p}}} \\ & \underbrace{+ m_i^{\text{solid}} \frac{1}{\beta_i} \sum_j m_j \frac{4\nu}{(\rho_i + \rho_j)} \frac{\mathbf{r}_{ij} \cdot \nabla W(r_{ij}, h_{ij})}{(|\mathbf{r}_{ij}|^2 + \eta)} \mathbf{u}_{ij}}_{\mathbf{f}_{i,\text{visc}}} \end{aligned} \tag{5.40}$$

where the summation index j is over all the fluid particles in the neighborhood of a solid particle indexed by i. We compute the coefficient of pressure drag $c_{d,\text{pressure}}$ and skin-friction drag $c_{d,\text{skin-friction}}$, and coefficient of lift c_l due to pressure by,

$$c_{d,\text{pressure}} = \frac{\mathbf{f}_\text{p} \cdot \mathbf{e}_x}{\frac{1}{2}\rho_0 U_\infty^2 L}, \quad c_{d,\text{skin-friction}} = \frac{\mathbf{f}_\text{visc} \cdot \mathbf{e}_x}{\frac{1}{2}\rho_0 U_\infty^2 L}, \quad c_l = \frac{\mathbf{f}_\text{p} \cdot \mathbf{e}_y}{\frac{1}{2}\rho_0 U_\infty^2 L}, \tag{5.41}$$

where L is the characteristic length of the simulation, U_∞ is the free stream velocity, $\mathbf{f}_\text{visc} = \sum_j \mathbf{f}_{j,\text{visc}}$ and $\mathbf{f}_\text{p} = \sum_j \mathbf{f}_{j,\text{p}}$ is the sum over all the dummy wall particles, and $\mathbf{e}_x$ and $\mathbf{e}_y$ are the unit vectors in the x and y directions respectively.

5.3.4 Time Integration

We use Predict-Evaluate-Correct (PEC) integrator to integrate the position $\mathbf{r}_i$, velocity $\mathbf{u}_i$, and pressure p_i. The integrator is as follows: to start with, predict the properties at an intermediate time value $n + \frac{1}{2}$,

$$\mathbf{u}_i^{n+\frac{1}{2}} = \mathbf{u}_i^n + \frac{\Delta t}{2}\frac{\tilde{\mathrm{d}}\mathbf{u}_i^n}{\mathrm{d}t}, \tag{5.42}$$

$$\tilde{\mathbf{u}}_i^{n+\frac{1}{2}} = \mathbf{u}_i^{n+\frac{1}{2}} + \frac{\delta\mathbf{r}_i^n}{\Delta t}, \tag{5.43}$$

$$\mathbf{r}_i^{n+\frac{1}{2}} = \mathbf{r}_i^n + \frac{\Delta t}{2}\tilde{\mathbf{u}}_i^{n+\frac{1}{2}}, \tag{5.44}$$

$$p_i^{n+\frac{1}{2}} = p_i^n + \frac{\Delta t}{2}\frac{\tilde{\mathrm{d}}\mathbf{p}_i^n}{\mathrm{d}t}, \tag{5.45}$$

then, estimate the new accelerations at $n+\frac{1}{2}$. Finally, correct the properties to get the corresponding values at the new time $n + 1$,

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + \Delta t\frac{\tilde{\mathrm{d}}\mathbf{u}_i^{n+\frac{1}{2}}}{\mathrm{d}t}, \tag{5.46}$$

$$\tilde{\mathbf{u}}_i^{n+1} = \mathbf{u}_i^{n+1} + \frac{\delta\mathbf{r}_i^{n+\frac{1}{2}}}{\Delta t}, \tag{5.47}$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t\tilde{\mathbf{u}}_i^{n+1}, \tag{5.48}$$

$$p_i^{n+1} = p_i^n + \Delta t\frac{\tilde{\mathrm{d}}\mathbf{p}_i^{n+\frac{1}{2}}}{\mathrm{d}t}. \tag{5.49}$$

The time-step is determined by the highest resolution used in the domain, and the minimum of the CFL criterion and the viscous condition is taken:

$$\Delta t = \min\left(0.25\left(\frac{h_{\min}}{U + c_s}\right), 0.125\left(\frac{h_{\min}^2}{\nu}\right)\right). \tag{5.50}$$

5.4 Adaptive Refinement

We provide a broad overview of the method before delving into the details. The adaptive refinement algorithm involves the following key ideas:

- A particle is split if its mass is greater than m_{max}. Note that m_{max} is space varying. The splitting is performed using the approach of Feldman et al. (2007) and Vacondio et al. (2013b). We normally split each particle into 7 child particles in two dimensions.
- A particle i is allowed to merge with another particle j if $r_{ij} < (h_i + h_j)/2$ and $m_i + m_j < \max(m_{max}[i], m_{max}[j])$. The merging algorithm is fully parallel and only particles that are mutually closest to each other are merged. That is, only if particle i's closest allowed merge particle is j and j's closest allowed merge particle is i, will particle i and j be merged. More details on the merging algorithm are provided below.
- When the particles are split they are iteratively merged three times in order to merge any split particles with nearby particles.
- The maximum mass and minimum mass at a particular location are set automatically using a reference m_{ref} parameter that is automatically computed based on the minimum specified resolution and a ratio C_r similar to what is done by Yang et al. (2019). $m_{max} = 1.05 m_{ref}$ and $m_{min} = 0.5 m_{ref}$.
- A collection of "background" points is used to adaptively set the minimum and maximum mass values of the fluid particles to control the adaptive resolution. If the body moves, this background is also updated.

Remark. Note that the last point is now redundant. The use of background particle is only to facilitate the refinement process and does not have any bearing on the scheme. We also show in the Section 5.5.7 that background particles can be eliminated and the minimum and maximum mass values can be obtained from the fluid position from the boundary, as long as it is in a internal flow.

The global minimum and maximum size of the particles is specified. Any solid bodies (barring the far-field slip walls) are assumed to be specified at the smallest size. The reference mass increases by the ratio C_r from the smallest particle to the largest. This produces a smooth increase in the number of particles in each region.

Algorithm 4 Splitting of the particles.

$\epsilon \leftarrow 0.4 \quad \alpha \leftarrow 0.9$
for all i of particles which are not fixed **do**
  if $m_i > m_{i,\max}$ **then**
    split particle i to 7 daughters
    for $k = 0;\ k < 7;\ k{+}{+}$ **do**
      $m_{i,k} \leftarrow \frac{m_i}{7}, \quad h_{i,k} \leftarrow \alpha h_i$
      $\mathbf{u}_{i,k} \leftarrow \mathbf{u}_i, \quad \tilde{\mathbf{u}}_{i,k} \leftarrow \tilde{\mathbf{u}}_i$
      $p_{i,k} \leftarrow p_i, \quad \rho_{i,k} \leftarrow \rho_i$
    $\mathbf{x}_{i,0} \leftarrow \mathbf{x}_i$
    for $k = 1;\ k < 7;\ k{+}{+}$ **do**
      $\mathbf{x}_{i,k} \leftarrow \mathbf{x}_i + \epsilon h_i \cos\left(\frac{k\pi}{3}\right)$

5.4.1 Adaptive Splitting

The algorithm for splitting particles follows that of Feldman et al. (2007) and Vacondio et al. (2012b). If a particle's mass is larger than the maximum allowed mass, m_{max}, then it is split into 7 particles. The original particle is called the parent particle and the split particles are called child particles. Six child particles are placed in a hexagonal arrangement with one child particle at the center as shown in fig. 5.1. The parent particle has a smoothing length of h, the child particles have a smoothing radius given by αh and they are placed on a circle of radius ϵh. These parameters α, ϵ are normally computed so as to minimize the density error as discussed in (Feldman et al. 2007; Vacondio et al. 2012b). We choose the parameters $\alpha = 0.9, \epsilon = 0.4$ for the equal mass ratio case. We note that in the present work these are only initial values of the distance and the smoothing length factors. After we split the particles we perform merging followed by shifting and a Taylor series correction. These corrections are accurate due the choice of the values of α and ϵ. Subsequently, we use the optimal smoothing length as discussed in the following. The mass of all the particles is the same and is equal to a seventh of the parent's mass. This configuration produces very little error. We use a quintic spline kernel for all computations in this work. The density, velocity, and pressure values of the parent particle are copied to the children. The children also copy the values of the m_{min}, and m_{max}.

We note that in Vacondio et al. (2012b), the value of parameter α is 0.9. This implies that the smoothing radius of the child particle is 0.9 times that of the parent despite it having a mass of

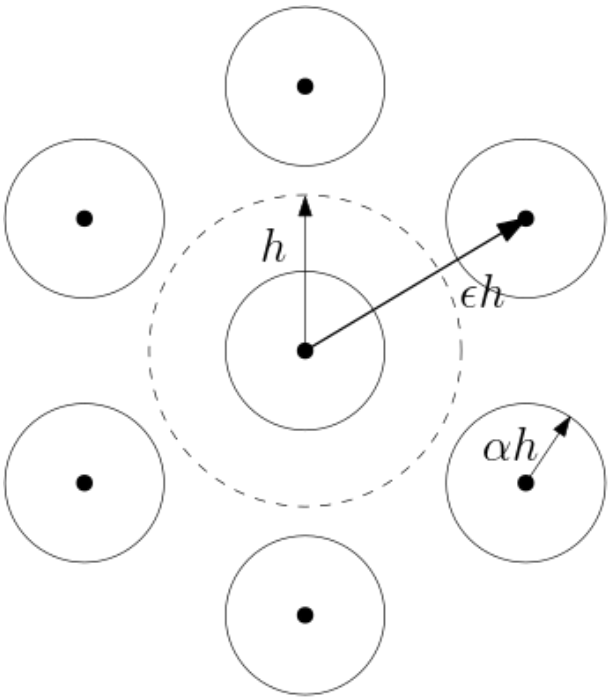


Figure 5.1: Sketch of particle splitting. The parent particle is shown as a dashed circle and has a smoothing length of h. Six child particles are placed in a hexagonal pattern with one child at the center.

around a seventh of the parent. Normally in an SPH simulation one tends to choose $m = \rho\Delta x^d$, where d is the number of spatial dimensions and Δx is the inter-particle spacing. Furthermore, $h = k\Delta x$ and k depends on the choice of the kernel. Thus, the value of $\alpha = 0.9$ is much larger than what one would ordinarily expect. This makes the original approach computationally inefficient and significantly increases the number of neighbors of each particle. This also reduces the accuracy of the method since the smoothing errors are larger. In the present work we find the average mass of particles in the neighborhood of each particle and use this to set the smoothing length using, $h = C(m/\rho)^{1/d}$, where C is a constant. In regions where the particle mass is uniform, this attains the ideal h value that would have been set without the use of adaptive resolution. This gives us an optimal h and is therefore computationally efficient. We test the accuracy of our method with a suite of benchmark problems in Section 5.5 and find that this does not affect the accuracy of the method.

The implementation of particle splitting is relatively straightforward. The adaptive splitting may be performed either every iteration or every $n_{adapt} > 1$ iterations. Any particles whose mass

is greater than the m_{max} value are split. Once these particles are identified, the total number of particles that need to be split can be identified. In addition, we also identify the particles that are to be merged as discussed in the next section. Hence, the total number of new particles that need to be created is known. The new child particles are then stored over any unused merged particles and new particles that have been created. Each of these steps are easy to implement in parallel using a combination of element-wise and reduction operations.

5.4.2 Merging Particles

The merging algorithm is in principle simple and we use essentially the same approach as discussed in (Vacondio et al. 2013b). We note that the smoothing radius of the particles is initially set as discussed in (Vacondio et al. 2013b). If we have two particles at locations $\mathbf{r}_a$, $\mathbf{r}_b$. The location of the merged particle is at,

$$\mathbf{r}_m = \frac{m_a\mathbf{r}_a + m_b\mathbf{r}_b}{m_m}, \tag{5.51}$$

where $m_m = m_a + m_b$ is the mass of the merged particle. The velocity is set using the mass-weighted mean as,

$$\mathbf{u}_m = \frac{m_a\mathbf{u}_a + m_b\mathbf{u}_b}{m_m}. \tag{5.52}$$

A similar form is used for any scalar properties like pressure. The position $\mathbf{r}_m$ and velocity $\mathbf{u}_m$ are obtained by ensuring the conservation of momentum. The smoothing radius is set by minimizing the density error (Vacondio et al. 2013b),

$$h = \left(\frac{m_m W(\mathbf{0}, 1)}{m_a W(\mathbf{r}_m - \mathbf{r}_a, h_a) + m_b W(\mathbf{r}_m - \mathbf{r}_b, h_b)}\right)^{1/d}, \tag{5.53}$$

where $W(\mathbf{x}, h)$ is the kernel function and d is the number of spatial dimensions.

Once the entire splitting and merging process is complete, the smoothing length h is set to an optimal value as discussed in the previous section using the average mass of the neighboring particles.

The parallelization of the merge step is however, a non-trivial problem which we discuss here. We wish to use a parallel algorithm that can identify possible merge candidates in one loop over the particle neighbors. The algorithm is designed so each particle can identify a suitable merge partner in parallel. This is achieved using the following approach.

- A particle i is allowed to merge with another particle j if the particle j has not been identified for splitting and $r_{ij} < (h_i + h_j)/2$ and $m_i + m_j < \max(m_{max}[i], m_{max}[j])$. All neighbors of particle i are searched and the closest particle index **`closest_idx`** that satisfies these criterion is identified. This is a completely parallel operation.
- If the i'th particle's **`closest_idx`** is j, and if the closest index of the j'th particle is i, then the two particles may be merged. Otherwise the particles are not merged.
- Once a pair of merging particles are identified, the particle with the smaller numerical index value is retained and the particle with the larger index is marked for deletion.

This algorithm is entirely parallel and can be implemented on a CPU or GPU very easily. In fact, these computations may be implemented easily in the context of a SPH calculation. After the identification is complete, one can easily identify the particles that need to be deleted or merged.

The above algorithm may run into pathological particle configurations which will not merge enough particles. However, we find that this does not happen in practice and the algorithm works effectively especially since the particles are constantly moving and are homogenized by the use of a particle shifting procedure.

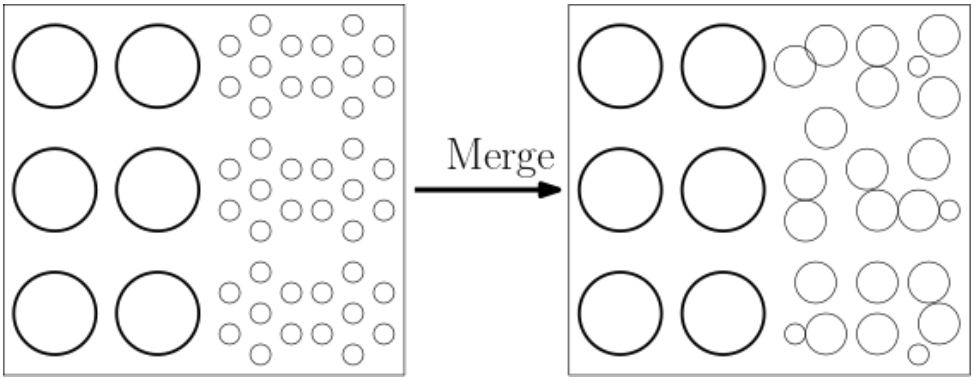


Figure 5.2: Sketch of particles splitting and then being merged. On the left side is a set of particles that split into 7 children each. On the right, these particles are merged to reduce the number of particles. Note that only one round of merging is complete at this stage.

It is important to note that when the particles are split, one particle is split into 7 (as discussed in Section 5.4.1). In order to reduce the number of particles we also iteratively perform merging

Algorithm 5 Merging of particles.

for all i which are not fixed **do**
  find neighbors of i
  if $m_i \leq m_{i,\max}$ **then**
    for j in neighbor indices **do**
      $m_{\text{merge}} \leftarrow m_i + m_j$
      $m_{\max,\min} \leftarrow \min(m_{i,\max}, m_{j,\max})$
      if $m_{\text{merge}} < m_{\max,\min}$ **&** j is closest of all neighbors **then**
        store index j for merging
for all i of particles which are to be merged **do**
  if merge pair of i is j and merge pair of j is i and $i < j$ **then**
    update $\mathbf{r}_i$ with its merged pair index using eq. (5.51)
    update $\mathbf{u}_i$ with its merged pair index using eq. (5.52)
    update h_i with its merged pair index using eq. (5.53)
  else
    if merge pairs match and $i > j$ **then**
      Delete particle i

using the same algorithm as discussed above. The reason we choose to split particles into 7 and then merge is that this tends to produce much lower errors since the particle distributions after splitting are more uniform and this makes it more effective to find merge partners. Since the merging is performed pairwise and we desire that on the average each particle be split into two particles, we must have at least three merges. Increasing the number of merges is computationally expensive so we limit it to three. After this, the smoothing length of all the particles is reset to the value corresponding to the average mass of the neighbors,

$$h_i = C\left(\frac{1}{\rho_0}\frac{\sum_j m_j}{N_i}\right)^{1/d}, \tag{5.54}$$

where C is a constant corresponding to the value of smoothing length factor used in the simulation. In our simulations we set this value to 1.2 for the quintic spline kernel.

In fig. 5.2, we show on the left two columns of parent particles that are moving. As they move to the right, they split into 7 children each. These are merged once to produce the particles on the right. With a subsequent merge the remaining small particles are also merged into larger

particles depending on the allowed maximum and minimum masses. The figure indicates that the particles are disordered. In order to correct these we perform particle shifting iteratively three times using eq. (2.26) and correct the properties of the fluid using eq. (2.28). In Section 5.4 we show some particle plots (see Figure 5.5) where one can clearly see that the particles are uniformly distributed.

5.4.3 Automatic Adaptation

The key part of the adaptive split and merge algorithm is in setting the appropriate $m_{max}(\mathbf{r})$ and $m_{min}(\mathbf{r})$ spatially. In simple cases, it is possible to manually assign the appropriate reference mass for different spatial regions. On the other hand for more complex cases we may not be able to set this manually. For example when simulating the flow past a bluff body, we would like to prescribe the minimum and maximum resolutions and automatically define the reference mass based on the distance from the solid body. In addition when the solid body moves, the reference mass should be suitably updated. Finally, the algorithm should also support solution adaptivity. We discuss the simpler case of geometry dependent spatial adaptation and then discuss how solution adaptivity can be added.

We setup the discussion in the context of wind-tunnel-like problems where a collection of stationary or moving solid bodies is placed in a stream of fluid with a suitable inlet and outlet. In these class of problems, the solid body typically defines the highest resolution since this is where the largest gradients are observed.

We use the term *size* of a particle to refer to the inter-particle spacing Δs. We determine a suitable reference mass in a region, m_{ref} and then set $m_{min} = 0.5m_{ref}$ and $m_{max} = 1.05m_{ref}$. The size of the particle immediately determines its m_{ref}, for in two dimensions, $m_{ref} = \rho(\Delta s)^d$ where d is the number of spatial dimensions. In order to smoothly vary the regions, we use a parameter $1.05 \leq C_r \leq 1.2$. The sizes of particles in two adjacent regions are in this ratio, i.e. $\Delta s_{k+1} = C_r \Delta s_k$, where k indicates a layer of particles with a similar resolution. We assume that the minimum resolution Δs_{min} and the maximum resolution Δs_{max} for the particles are known quantities. We note that $h = 3\Delta s$ for the simulations in this work.

We employ a set of stationary background particles to define the reference mass. These background particles are not involved in the computation of the governing equations of motion of the fluid or solid. They are merely used to set m_{ref} based on the requirements. The background particles are initially setup with a constant size of Δs_{max}. The solid geometry of interest is dis-

cretized at a resolution of Δs_{min}. Given these, we initialize the background when the simulation starts as follows.

Remark. Note that the background particles are used in the simulation to help aid in setting the mass ratios correctly on the fluid particles. We run the `UpdateSmoothingLength` algorithm (Section 5.5.7) on the background particles to set the adaptive zones and use these background particles as references to the fluid particles to recognize the zones. We later realized that all of this could be done directly on the fluid particles, but the problem was with the initialization before the simulation started. To obtain a smoother distribution of fluid particles before the simulation start, we retained the background particles and used them to initialize the fluid particles. After the initialization the background particle set up can be removed and the fluid particles themselves can be used to define the reference mass. The `UpdateSmoothingLength` algorithm is executed directly on the fluids, resulting in improved performance and reduced memory usage. This is shown with examples, both stationary and moving, which demonstrate that the fluid dynamics remain unaffected. Hence, we can replace the background particles with fluid particles in the subsequent discussion.

1. Iterate over all the background particles. If a background particle has a solid particle as a neighbor, then the background particle is marked as being near a boundary. In our implementation, we have a simple integer mask which is set to the value 1. These particles are set to have the smallest size (the same as that of the solid particles). We call these the *boundary background particles*.

2. Once the boundary background particles are identified, we iterate over the remaining particles and find the minimum (Δs_{min}), maximum (Δs_{max}), and geometric mean (Δs_{avg}) of the sizes of the neighboring particles. If the $\Delta s_{max}/\Delta s_{min} < C_r^3$, this suggests that the regions are near the ideal distribution, and we set the size of the particle to be equal to $C_r \Delta s_{min}$. If $\Delta s_{max}/\Delta s_{min} >= C_r^3$, then we set the size of the particle to Δs_{avg}. This allows the particle sizes to be refined rapidly in the initial stages when most of the particles are at the highest resolution. When the distribution is nearing the desired distribution we ensure that the nearby layers are such that $\Delta s_{k+1} = C_r \Delta s_k$, where k indicates a layer. We note that the size of the particle immediately determines m_{ref}, m_{min}, and m_{max}.

3. Once the reference mass of the particles is set, the particles are split if required using the same splitting algorithm as used for the fluid particles.

4. The smoothing length of the particles is now set such that the number of neighbors is roughly the same. Equation (5.55) is used for this purpose and is discussed below. The background particles are also moved to distribute them uniformly using the same PST method as used for the fluid. Both the method of (Vacondio et al. 2013b) or (Lind et al. 2012) work well although we use that of (Lind et al. 2012) in this work since it is parameter free and works very well. These two operations of setting the smoothing length and using a PST are repeated three times.

The equation used to iteratively set the smoothing length of the background particles is the same as that used in (Yang et al. 2019) and is reproduced here,

$$h_i^{n+1} = \frac{1}{2}\left(\frac{h_i^n}{2}\left(1 + \sqrt{\frac{N_r}{N_i^n}}\right) + \frac{\sum_j h_j^n}{N_i^n}\right), \tag{5.55}$$

Algorithm 6 Update spacing of particles

```
Define Δs for all particles
for all i of particles which are not fixed do
    find neighbors of i
    find Δs_min, and Δs_max in the neighborhood
    find average spacing Δs_avg in the neighborhood
    if Δs_max / Δs_min < (C_r)^3 then
        h_tmp ← min(Δs_max, C_r Δs_min)
    else
        h_tmp ← Δs_avg
for all i of particles which are not fixed do
    m_i ← ρ_i h_tmp^2
    m_i,max ← 1.05 m_i    m_i,min ← 0.5 m_i
```

where N_r is a reference number of neighbor particles, N_i^n is the number of neighbors for particle i at iteration level n. This approach ensures that each particle has close to N_r neighbors eventually. We reiterate that this algorithm is only used for the background particles so that they smoothly vary and the computations of the references masses are smooth. For the two-dimensional flow problems considered here, we use $N_r = 48$.

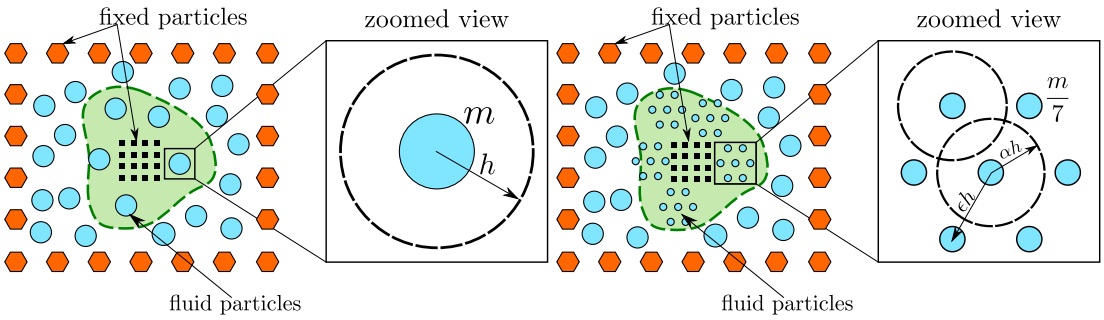


Figure 5.3: Illustration of the splitting algorithm (a) before the split, and (b) after the split. The particles outside the green zone have mass m and the maximum mass is $m_{\max} = m$. All the particles inside the green region have maximum mass $m_{\max} = m/2$. After the split the particles undergo merging which is not illustrated.

The above steps are repeated $3\lceil \log(\Delta s_{\max}/\Delta s_{\min})/\log(C_r) \rceil$ times to setup the initial background particles. The fluid particle resolution is set by finding the minimum of the background particle reference mass in its neighborhood. Thus the background particles only define the spatial resolution for the fluid particles.

Figure 5.4 we show the background particles and the corresponding number of split levels with 0 being the smallest size particles and 7 being the largest. In this case the minimum particle spacing is 0.1 and 0.4 is the maximum spacing. We choose a $C_r = 1.2$ and this generates roughly 8 regions with differing m_{ref} values. The fluid particles created for this distribution of particles is shown in fig. 5.5. Here we can see that there are only 4 layers since when particles split they effectively split when the mass from one layer to the next jumps by a factor of two.

When the solid bodies move, the algorithm above is executed once every few iterations. This automatically adapts the reference mass distribution in a smooth fashion. Since the motion of the bodies in each time step is typically a fraction of the local smoothing length, we only need to perform one iteration of the above.

In fig. 5.6 we show the case of two unit square solids placed in a fluid, the background particles are shown and the particle size is smoothly decreasing towards the solid geometry. We move each square by 0.05 units away from each other in each step and update the background by performing the steps discussed above once each time step. We do this 60 times and the resulting background particles are shown in fig. 5.7. As can be clearly seen, the background mesh adapts

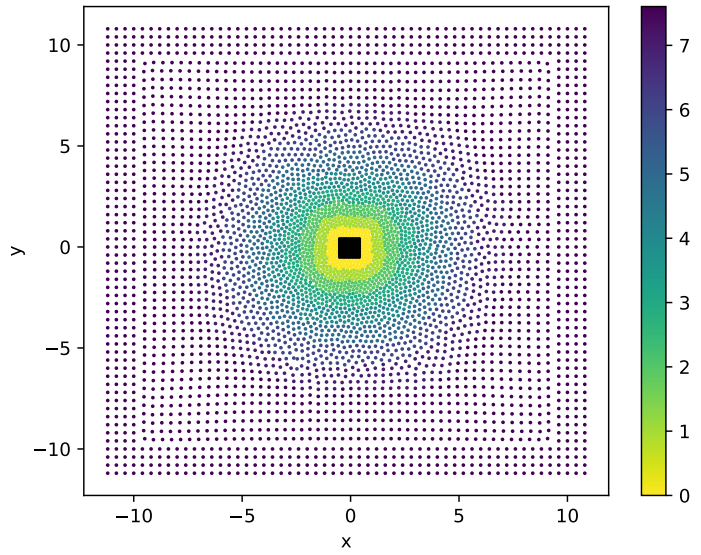


Figure 5.4: Background particle distribution for flow around a unit square shown in black.

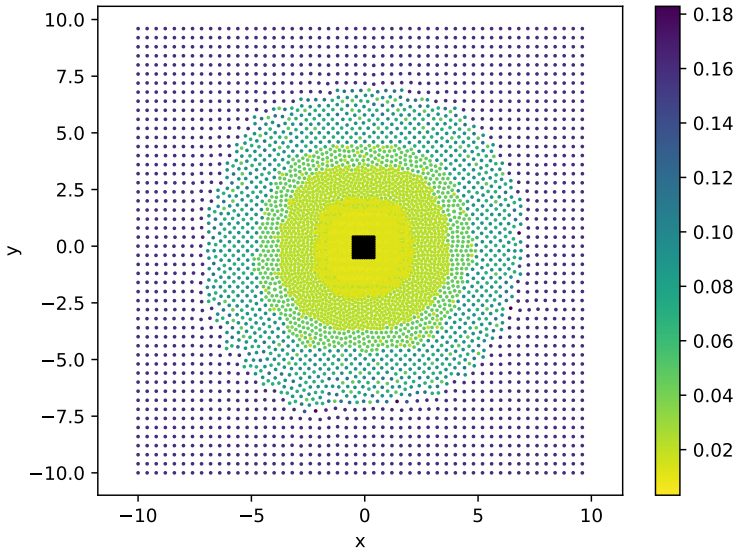


Figure 5.5: Corresponding fluid particles initialized using the background particles with the colors indicating the particle mass. The square solid is shown in black.

to the moving solid. This shows that the algorithm can comfortably handle moving geometries.

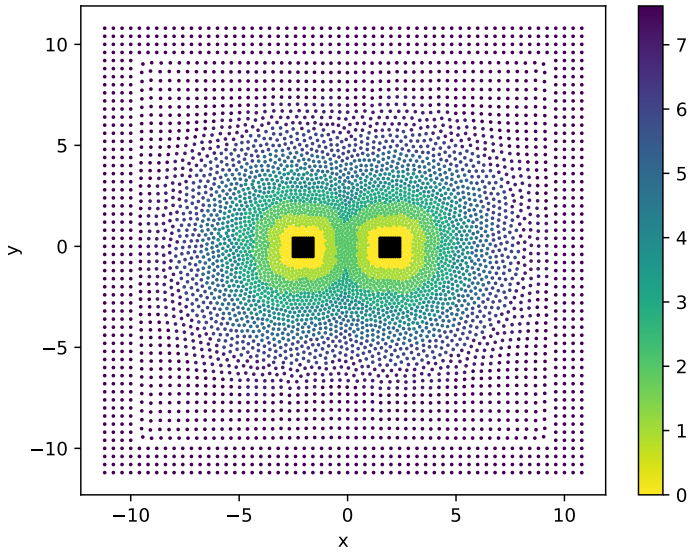


Figure 5.6: Background particle distribution with two square solid bodies shown in black.

We note that the current method may also be used to setup a specific user-defined region with a desired resolution. This is done by creating a set of particles that serve as a solid body but are only used to set the boundary background particles. These particles do not participate in any fluid-solid computations. Thus the approach offers a convenient way to define user-specified regions with different resolutions.

While we do not explore this extensively in the current work, it is easy to incorporate solution adaptivity using this framework. Let us assume that there is some solution dependent scalar $\phi(\mathbf{r})$ that may be evaluated using the fluid particles (like the magnitude of the vorticity) and are interpolated onto the background particles. We can use a linear mapping between the range of the values of ϕ to the minimum and maximum allowed resolution. Once the boundary background particles are identified (step 1 in the algorithm for the background particle) we use the ϕ value to appropriately set the resolution. The rest of the algorithm then proceeds as before to update the remaining particles.

We show examples of solution adaptivity based on the vorticity, ω in Section 5.5.5. In this case, we compute the absolute magnitude of the vorticity of the fluid particles and interpolate them onto the background particles as the value of ϕ. Any particles with a value of $\phi > k \max(\omega)$,

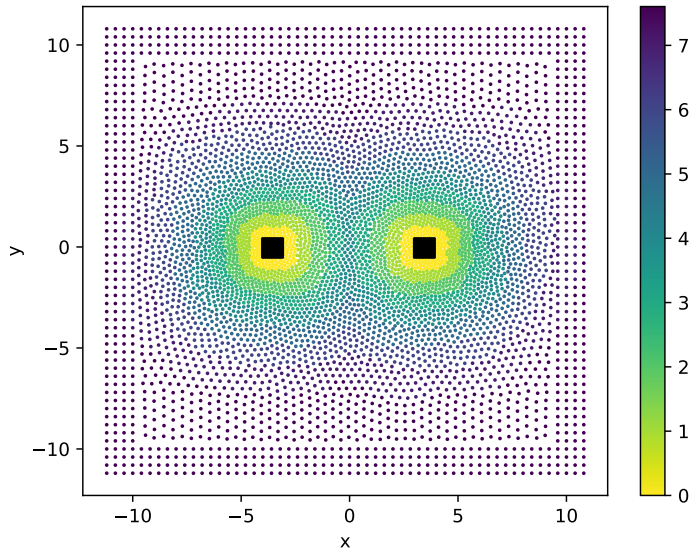


Figure 5.7: Background particle distribution with two square solid bodies after they move by a distance of 3 units in 60 steps.

where k is a user-specified value, are assigned the highest resolution. This approach allows us to track the vorticity adaptively. The approach may be easily extended to use different refinement criterion if so desired. The proposed algorithm can thus handle a variety of different forms of adaptivity.

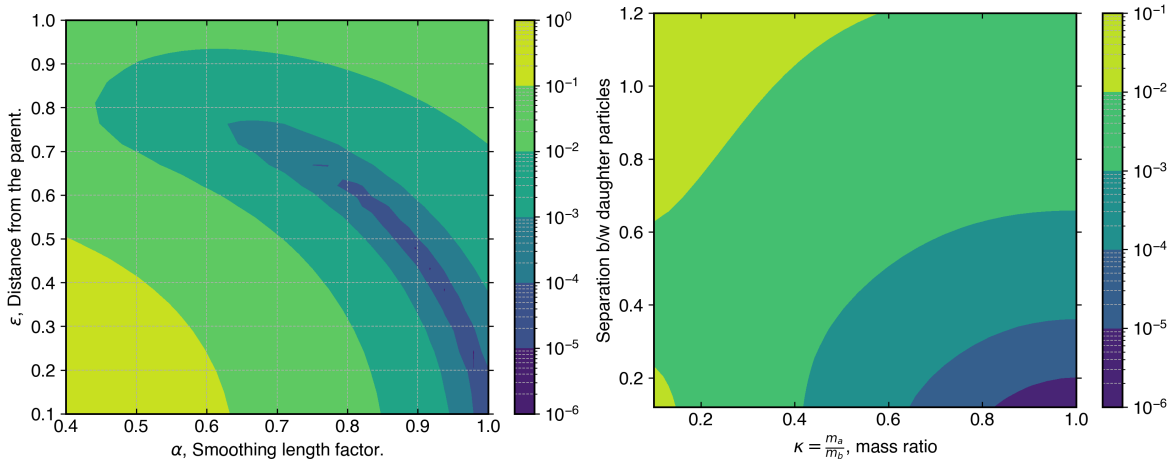
We want to assess the errors due the splitting and merging. We being with estimating the error in the global density due to merging of particles of different mass ratios, κ, at varying separation distance, for which, consider a square domain Ω of uniformly discretized points and two additional particles which are merging. Then, we compute the error in the global density (Vacondio et al. 2012b), due to the merging, defined as,

$$\mathcal{E}_m(\mathbf{x}) = \int_\Omega \left[(m_a + m_b)W_m(x, h_m) - m_a W_a(x, h_a) - m_b W_a(x, h_a)\right] \mathrm{d}\mathbf{x} \tag{5.56}$$

where h_m is set using eq. (5.53), subscripts a and b denote particles which are being merged, and subscript m denotes the merged particle, $\kappa = \frac{m_a}{m_b}$ is the mass ratio of the particles being merged. The integral in eq. (5.56) is evaluated numerically on a mesh of uniformly distributed points.

Figure 5.8b shows the error in the global density for two particles merging at varying separation distance and mass ratio. It is clear from the figure that the merge errors are low when the

particles are closer and have nearly equal mass.



(a) (b)

Figure 5.8: (a) Error in density by splitting one particle into 7 daughter particles with equal mass. We vary the smoothing length factor α and the distance of the 6 children particles from the parent particle. (b) Error in density while merging two particles of varying mass ratios, κ, and separation distance.

Next, following Vacondio et al. (2012b) in order to assess the error due to splitting one particle into 6+1 daughter particles, we compute the error in global density. We consider a square domain Ω of uniformly discretized points and a single SPH particle. We split a single SPH particle into 7 daughter particles, where 6 daughter particles are placed on the vertices of a hexagon centered around the parent particle and one daughter particle at the location of the parent particle. The distance of the 6 daughter particles from the center is controlled by the parameter ϵ, where $r_p = \epsilon h_p$, and the smoothing length of the k$^{\text{th}}$ daughter particle, $h_k = \alpha h_p$, is controlled by the parameter α. The global density error is then computed by evaluating,

$$\begin{aligned}\mathcal{E}_s(\mathbf{x}) = m_p^2\Big[\int_\Omega W_p^2(\mathbf{x}, h_p)\mathrm{d}\mathbf{x} - 2\sum_{k=1}^{7}\int_\Omega \lambda_k W_p(\mathbf{x}, h_p) W_k(\mathbf{x}, h_k)\mathrm{d}\mathbf{x} \\ + \sum_{k,l=1}^{7}\int_\Omega \lambda_k\lambda_l W_k(\mathbf{x}, h_k) W_l(\mathbf{x}, h_l)\mathrm{d}\mathbf{x}\Big],\end{aligned} \tag{5.57}$$

where λ_k is the mass ratio of the parent particle to the k^{th} daughter particle, the subscript p denote parent particle, the summation is over all the children particles. The kernel function is

computed and the integration is performed over the uniformly discretized points. The parent and all the daughter particles always have full kernel support. We take $\lambda_k = 1/7$ in this work based on the results of the errors in merging shown above.

Figure 5.8a shows the error in the global density for a particle split into 7 equal mass daughter particles. Based on the results of errors in merging and splitting we choose the values of the daughters' smoothing length factor $\alpha = 0.9$ and position from the center $\epsilon = 0.4$.

5.4.4 Algorithm

In this section we summarize the adaptive resolution algorithm. We start with a given solid body or multiple such bodies that are discretized at the highest desired resolution, with particle spacing, Δs_{min}. For complex geometries, we may use the particle packing method proposed in (Negi et al. 2021c) to generate uniformly distributed particles for discretizing the solid bodies. We prescribe a coarsest resolution Δs_{max} as well as the desired C_r factor which is typically between the values of 1.05 to 1.2. This effectively determines the width of each refinement layer. One may also manually specify the constraints on the mass in different spatial regions. Finally, we are given a domain of interest; in the problems considered in this work, the domain size is fixed and known a priori. Given this, the algorithm proceeds as follows.

1. We first initialize the background particles as discussed in Section 5.4.

2. Using the background particles, we initialize the fluid particles at the initial time. This is done by splitting and merging the particles based on the reference mass of the background particles. This is discussed in detail in Sections 5.4.1 and 5.4.2. The particle shifting algorithm of Lind et al. (2012) is applied at each stage to get a smooth distribution of particles.

3. The initialized fluid particles along with the given solid particles are then used to simulate the governing equations using an appropriate scheme. In the present work we use a highly modified EDAC-SPH scheme as discussed in Section 5.3.

4. At the end of every iteration, we find the nearest fluid particles to the background particles and set the reference mass of the fluid particles.

5. The fluid particles are adaptively split and merged every n_{adapt} iterations as discussed in Sections 5.4.1 and 5.4.2. Typically we choose n_{adapt} to be between 1 to 10. Similarly, the

background particles are updated (to accommodate moving bodies or solution adaptivity) every n_{bg} iterations. When the adaptation is entirely spatial and the solid boundaries do not move, the background does not need to be updated at all; we usually set this to 100 or 500 iterations. This parameter is adjustable depending on the requirements.

6. After merging, we shift the particles three times using eq. (2.26) and correct the properties of the fluid eq. (2.28).

7. The smoothing length of all the particles is set using the average mass of the neighbors as discussed in Section 5.4.1.

After the initialization is complete, the time-marching procedure of the SPH scheme is started. Before every iteration, the following is performed: (i) we update the background particles every n_{bg} iterations; (ii) thereafter, every n_{adapt} iterations, we split, merge, shift and correct the properties of the particles; (iii) we update the smoothing length of the particles based on average mass and update the nearest neighbor search algorithm; (iv) finally, we execute each time-step of the time-marching scheme as discussed in the Section 5.3.4.

In order to assess the accuracy of the algorithm, we consider the following test case. We consider a square region of side two units on which we discretize the function $f(x, y) = \sin(2\pi x) + \cos(2\pi x)$ using particles of equal size. The inner square region of unit side is split into particles. We consider two cases, the first using the formulation of Vacondio et al. (2012b) where each particle in the inner region is split into 7 daughter particles, and the second using the proposed method. We then compute the error in function and gradient approximation figs. 5.9 and 5.10. The results are summarized in table 5.1. As can be seen the proposed algorithm is accurate and has far fewer neighbors than Vacondio et al. (2012b). The proposed algorithm as accurate as Vacondio et al. (2012b) in the function approximation but the gradient of the function is slightly less accurate in L_∞ norm.

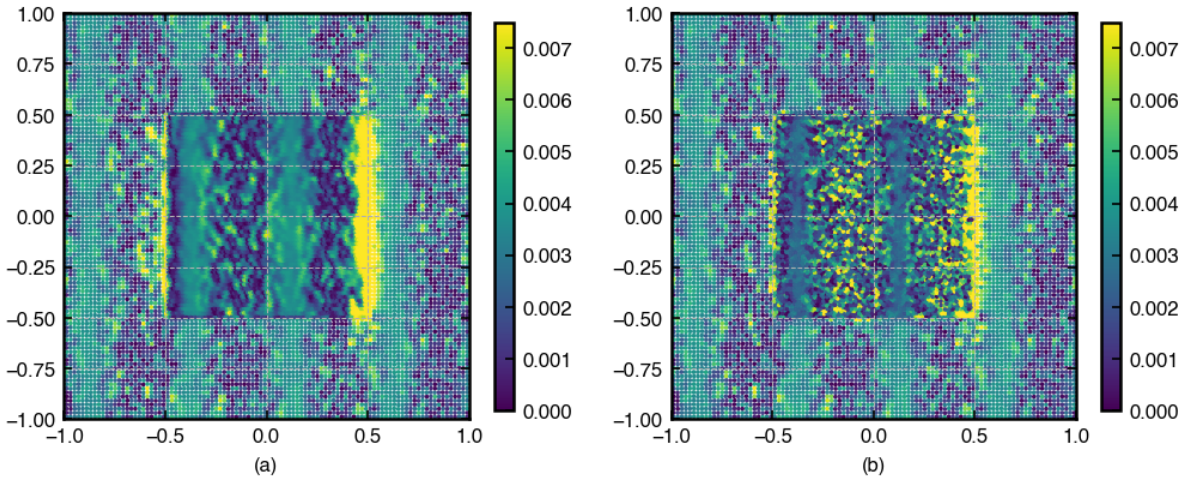


Figure 5.9: Error in approximation of a function $f(x) = \sin(2\pi x) + \cos(2\pi x)$ using (a) Vacondio et al. (2012b) formulation and (b) the proposed formulation.

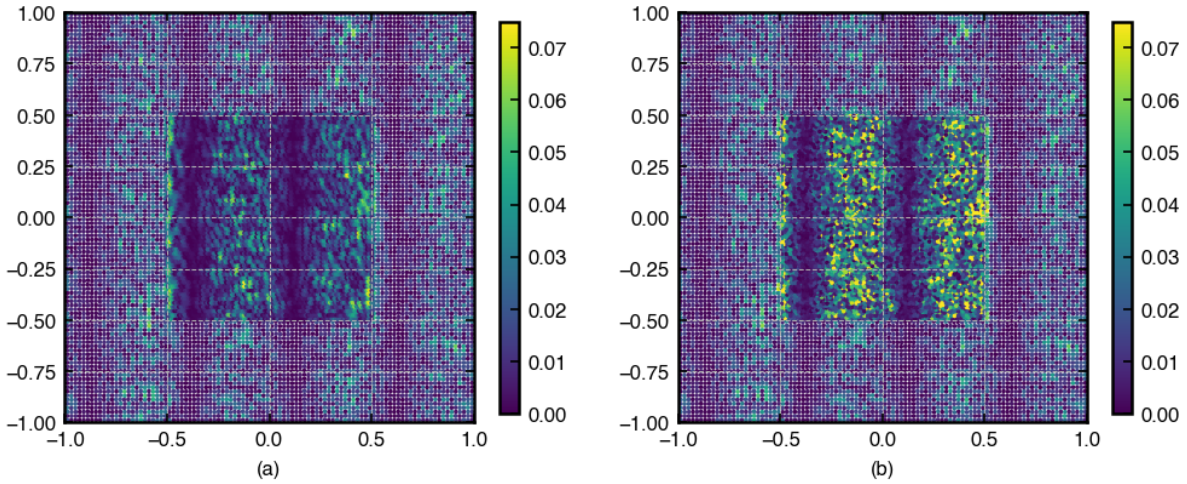


Figure 5.10: Error in approximation of the gradient of a function $f(x) = \sin(2\pi x) + \cos(2\pi x)$ using (a) Vacondio et al. (2012b) formulation and (b) the proposed formulation.

Table 5.1: Comparison of errors in L_∞ and L_1 norm of the proposed formulation and the formulation of Vacondio et al. (2012b).

	Vacondio et al.	Our Algorithm
L_∞ error in ρ	2.00e-15	1.11e-15
L_∞ error in f	1.0e-02	1.0e-02
L_1 error in f	2.5e-03	2.4e-03
L_∞ error in $\frac{\partial f}{\partial x}$	5.7e-02	1.2e-01
L_1 error in $\frac{\partial f}{\partial x}$	7.3e-03	8.0e-03
Average no. of neighbors	116	36

5.5 Results and Discussion

We apply the adaptive resolution technique proposed in this work to the test cases shown below. To start with, we apply our method to the classical numerical test cases with varying Reynolds numbers and compare with established results in the literature. We then simulate the flow past a circular cylinder at Reynolds numbers 40, 550, 1000, 3000, and 9500. We compare the results of our method to the high resolution vortex method results of Koumoutsakos et al. (1995), and Ramachandran (2004). We also show the results of solution-based dynamic particle resolution, where the particles are adaptively resolved to the highest resolution based on the magnitude of vorticity in the flow, for the flow past a circular cylinder and the flow past a C-shape at $Re = 2000$.

5.5.1 Taylor-Green Vortex

The Taylor-Green vortex is a widely used benchmark to study accuracy in SPH (Chiron et al. 2018; Vacondio et al. 2012b). The exact solution for the Taylor-Green vortex is given by,

$$u^+ = -U_\infty e^{bt} \cos(2\pi x^+/L) \sin(2\pi y^+/L), \tag{5.58}$$

$$v^+ = \quad U_\infty e^{bt} \sin(2\pi x^+/L) \cos(2\pi y^+/L), \tag{5.59}$$

$$p^+ = -U_\infty^2 e^{2bt} (\cos(4\pi x^+/L) + \cos(4\pi y^+/L))/4, \tag{5.60}$$

where the superscript $(.)^+$ indicate dimensional quantity, $U_\infty = 1$ m/s, $b = -8\pi^2/Re$, $Re = UL/\nu$, and $L = 1$ m. In the results we use the dimensionless velocities $u = u^+/U_\infty$ and $v = v^+/U_\infty$, pressure $p = p^+/\rho_0 U_\infty^2$, and distance $x = x/L$ and $y = y/L$. The domain is a square region

$[0, L] \times [0, L]$ with an inner refinement zone $[0.1L, 0.5L] \times [0.1L, 0.5L]$ where the resolution is 2 times higher than the outer resolution. We simulate the problem using the parameters given in table 5.2.

Table 5.2: Parameters used for the Taylor-Green vortex.

Quantity	Values
L, length of the domain	1 m
Time of simulation	2.5 s
c_s	10 m/s
ρ_0, reference density	1 kg/m^3
Reynolds number	200 & 1000
Resolution, $L/\Delta x_{\max} : L/\Delta x_{\min}$	[50 : 100] & [100 : 200] & [150 : 300]
Smoothing length factor, $h/\Delta x$	1.0

We consider two Reynolds numbers 200 and 1000, and simulate the problem using the adaptive algorithm proposed in this chapter with three different minimum resolutions $L/\Delta x_{\max}$ of 50, 100 and 150. We compare the results with the exact solution and the non-adaptive simulation with a resolution that matches the minimum resolution of the adaptive case. For the $Re = 200$ case we also compare with the Adaptive Particle Refinement (APR) results of (Chiron et al. 2018).

Figure 5.11 shows the particle plots indicating velocity magnitude and pressure for $Re = 200$ at $t = 2.5$ s. The velocity and pressure contours show less decay than (Chiron et al. 2018). Specifically, note that the high velocity regions near the interface of the layers are well maintained in the present work. Figure 5.12 shows the results for $Re = 1000$. It can be seen that the results are as expected and the contours do not show any decay or change in shape. We note that there is a drift in the average pressure and subtract it from the pressure contour plots. This drift in pressure is due to errors in the total volume conservation (Sun et al. 2019a), which is computed as follows,

$$\epsilon_V(\%) = \left| \frac{\sum_i m_i/\rho_i}{L^2} - 1 \right| \times 100. \tag{5.61}$$

Figure 5.13 shows the error in total volume. We compare the adaptive and non-adaptive cases with the non-adaptive δ^+-SPH results by (Sun et al. 2019a) who simulated the problem with

$Re = 1000$ at a resolution $L/\Delta x = 400$. As can be seen in Section 5.5.1, the errors in the present work for $L/\Delta x = 150$ are almost five times smaller than the case with $L/\Delta x = 400$ of (Sun et al. 2019a). However, the adaptive cases have a larger errors which drop as the resolution increases.

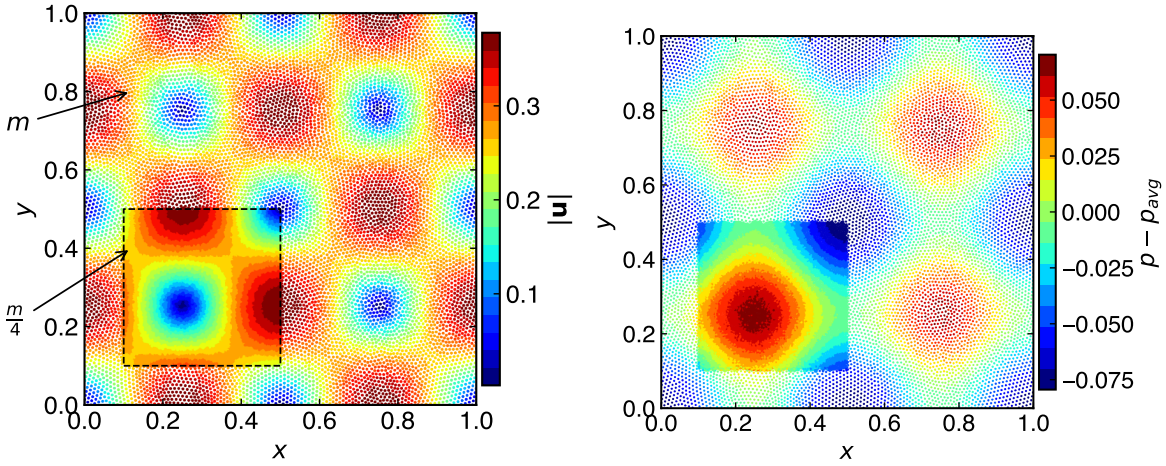


Figure 5.11: Particle plots for the Taylor-Green vortex at $t = 2.5$ s and $L/\Delta x_{\max} = 100$. Reynolds number is 200. The mass of the particles inside the dashed region is $1/4$ times the mass of the particles outside the region i.e., inside the region the resolution $L/\Delta x$ is 200.

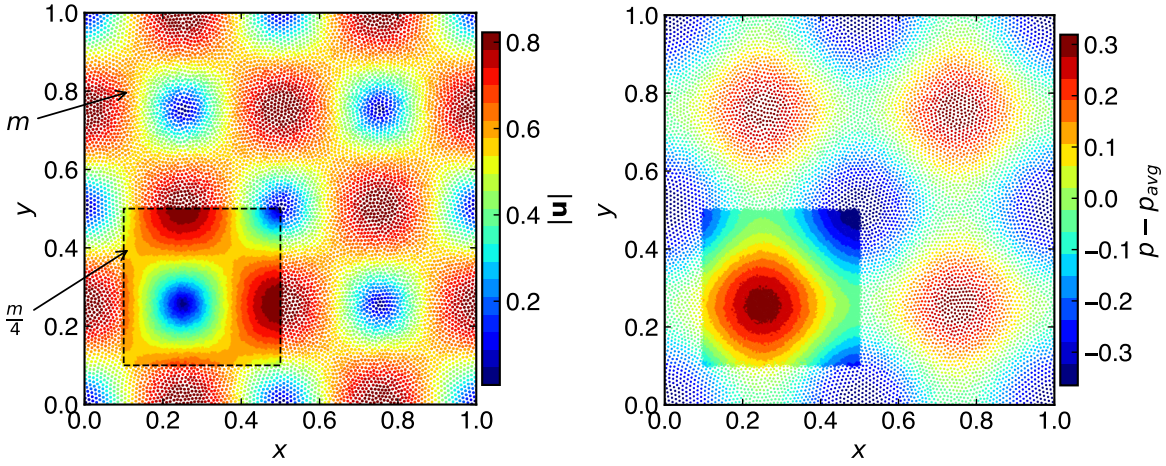


Figure 5.12: Particle plots for the Taylor-Green vortex at $t = 2.5$ s and $L/\Delta x_{\max} = 100$. Reynolds number is 1000. The mass of the particles inside the dashed region is $1/4$ times the mass of the particles outside the region indicating the resolution inside corresponds to $L/\Delta x$ of 200.

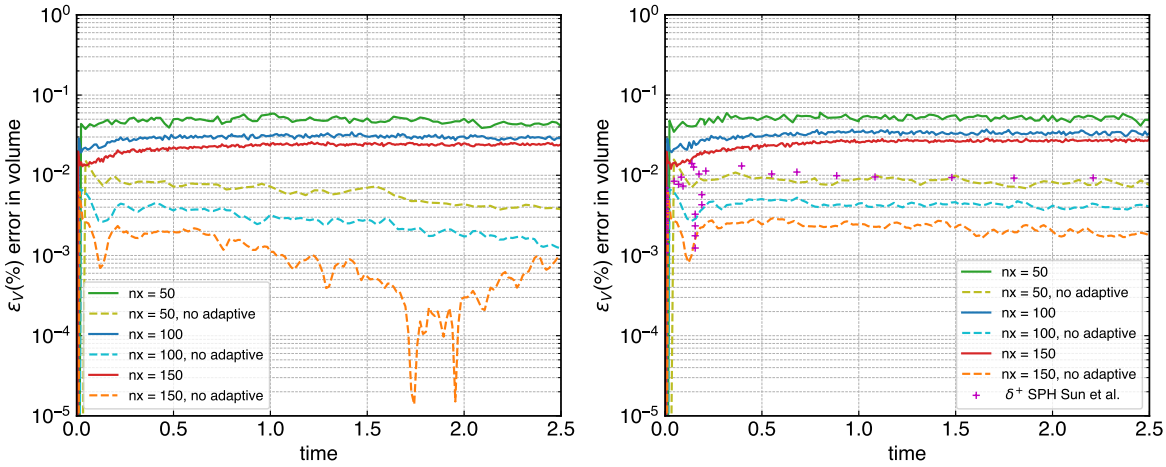
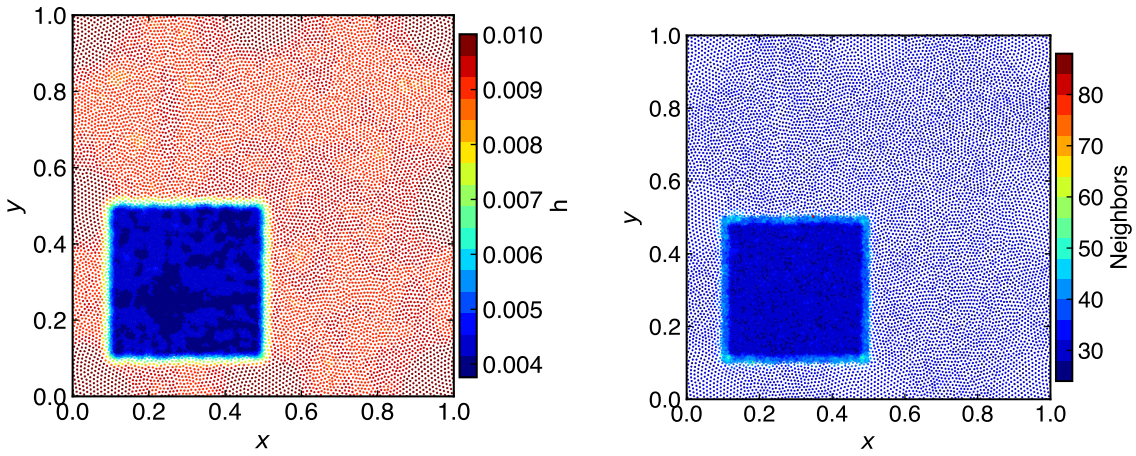


Figure 5.13: Evolution of the percentage change in total volume occupied by the particles from the initial one, $\epsilon_V(\%)$, for the Reynolds number 200 (left) and 1000 (right).

Figure 5.14a shows the spatial distribution of the smoothing length h. As can be seen the smoothing length is almost constant in the interior of the respective regions. At the interface between the two regions having different mass the value is changing gradually. Figure 5.14b shows the distribution of number of neighbors of each particle. It can be seen that in the interior of the regions the value is around 30. Whereas, in the interface between the two regions it is larger as would be expected. Since bulk of the particles have minimum number of neighbors the method is efficient.

In fig. 5.15 and fig. 5.16 we plot the maximum velocity decay and the L_1 error in the velocity for $Re = 200$, and $Re = 1000$ respectively with different minimum resolutions. The maximum velocity decay shows good agreement with the exact solution. We also compare with the non-adaptive case at different resolutions. Although we do not expect greater accuracy than the non-adaptive case due to the presence of lower resolution regions, we expect the errors to be of the same order as that of the non-adaptive case. The L_1 norms reveal that the errors in the adaptive case are almost 2 times the errors in the non-adaptive case for $Re = 200$, whereas for the $Re = 1000$ case, the errors are lower. The increase in the L_1 error for the adaptive case at $Re = 200$ is due to the effect of having an approximately constant number of neighbors in the adaptive region. Normally, the discretization error due to the viscous operator reduces when the number of neighbors is increased. In addition, this error is significant at low Reynolds numbers and hence causes an increase in the L_1 error. This is consistent with the findings of (Negi et al.



(a) Smoothing length, h.

(b) No. of neighbors.

Figure 5.14: Particle plots for the Taylor-Green vortex shown in fig. 5.12. In (a) the distribution of the smoothing length h is shown, and in (b) the number of neighbors of each particle are shown.

2021a) (figure 11 and table 5), and (Negi et al. 2021d) (figure 15). A theoretical proof of this is available in the work of (Fatehi et al. 2011). At $Re = 200$ the viscous error dominates, however, at $Re = 1000$ the viscous effect is not dominant and the L_1 error behaves as expected; the adaptive method has a smaller L_1 error than the non-adaptive case.

In fig. 5.17 we show the kinetic energy decay for $Re = 200$ and $Re = 1000$ at different minimum resolutions. We compare with the exact, non-adaptive, and the APR simulation of (Chiron et al. 2018). Our results match the non-adaptive decay and barring a slight increase the exact decay, whereas the APR scheme shows comparatively large decay. Figure 5.18 shows the error in the kinetic energy versus time. For the $Re = 200$ case the results show an anomalous behavior as seen from the L_1 error. The $Re = 1000$ cases behave as expected.

The above results show that the proposed method is accurate, displays less dissipation than other recent techniques proposed for adaptive resolution, and requires minimum number of neighbors for bulk of the particles. This makes the proposed method both accurate and efficient.

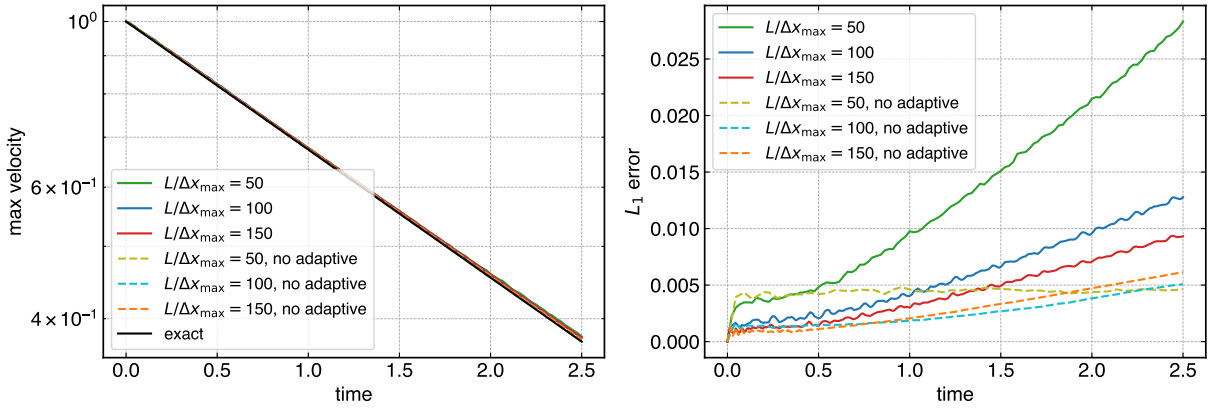


Figure 5.15: Decay of the maximum velocity (left) and L_1 error in the velocity (right) for the Taylor-Green vortex at $Re = 200$.

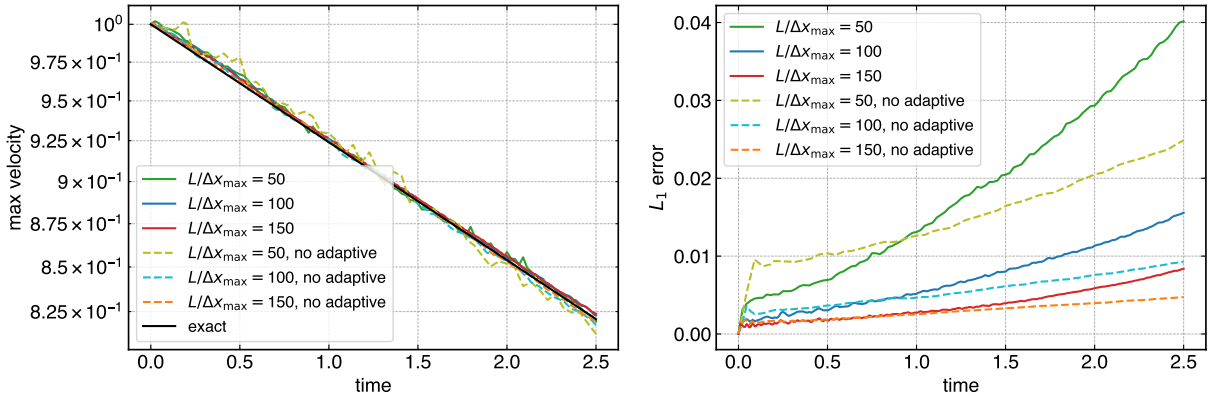


Figure 5.16: Decay of the maximum velocity (left) and L_1 error in the velocity (right) for the Taylor-Green vortex at $Re = 1000$.

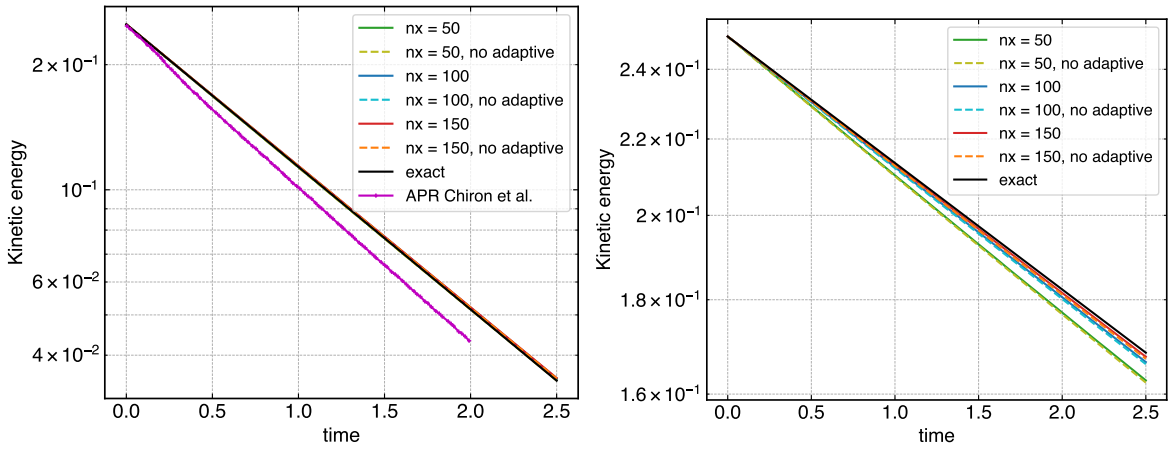


Figure 5.17: Kinetic energy decay of the Taylor-Green vortex at $Re = 200$ (left) and $Re = 1000$ (right). We use the exact, non-adaptive, and APR scheme (only for $Re = 200$) (Chiron et al. 2018) for comparison.

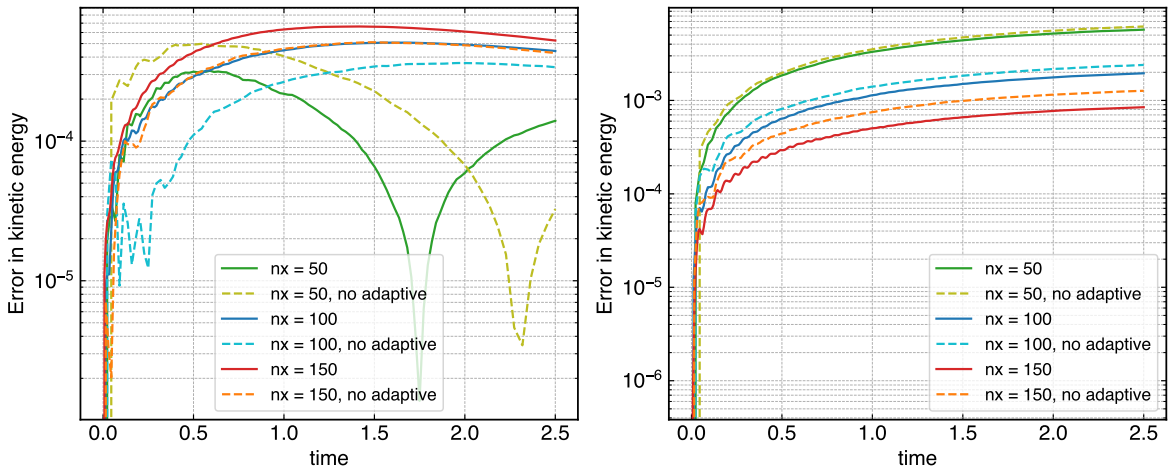


Figure 5.18: Error in kinetic energy decay of the Taylor-Green vortex at $Re = 200$ (left) and $Re = 1000$ (right).

5.5.2 Gresho-Chan Vortex

Gresho-Chan vortex (Gresho et al. 1990) is a two-dimensional inviscid numerical test case with periodic boundary conditions in both the x and y directions. This problem tests the numerical stability of the method, and the conservation properties. Considered as a difficult test case (Rosswog 2015), this test case is widely used by the astrophysical community (Hopkins 2015; Liska et al. 2003; Springel 2010b). The problem is of a rotating vortex inside a domain of unit length where the centrifugal force due to azimuthal velocity balances the pressure gradient. The distances are non-dimensionalized by the domain length L. The initial radial velocity is zero, and the azimuthal velocity in non-dimensional form is given by,

$$u_\varphi(r) = \begin{cases} r/R & \text{for } 0 \le r < R, \\ 2 - r/R & \text{for } R \le r < 2R, \\ 0 & \text{for } r \ge 2R, \end{cases} \tag{5.62}$$

where $R = 0.2L$, and r is the distance from the center of the vortex located at the origin $(0, 0)$. The artificial speed of sound is $c_s = 10$ m/s, and the smoothing length factor, $h/\Delta x = 1.0$. The non-dimensional pressure, balanced by the centrifugal velocity, is given by,

$$p(r) = p_0 + \begin{cases} \frac{25}{2}r^2 & \text{for } 0 \le r < R, \\ 4 - 4\log(0.2) + \frac{25}{2}r^2 - 20r + 4\log(r) & \text{for } R \le r < 2R, \\ 4\log(2) - 4 & \text{for } r \ge 2R, \end{cases} \tag{5.63}$$

where $p_0 = 5$ is the reference pressure. We adaptively refine a semi-circular region of radius 0.45 around the origin with particles of mass around 0.5 times that of the outer particles. We simulate the problem for $t = 3$ s. We compare our results with the exact solution and the non-adaptive cases. We consider two different minimum resolutions $L/\Delta x_{\max}$ of 50 and 100. The particles are initially placed on a uniform Cartesian grid. Figure 5.19 shows the particle positions at $t = 3$ s. It is difficult to assess the difference between the simulations from this result. Figure 5.20 shows the magnitude of the velocity of all the particles in the domain as a function of the distance r from the center of the vortex. The red-line indicates the exact velocity magnitude. The figure also indicates the L_1 norm of the error, which is computed as,

$$L_1 = \frac{1}{N}\sum_{i=1}^{N} |\mathbf{u}_i - \mathbf{u}_{\text{exact}}(\mathbf{r}_i)|. \tag{5.64}$$

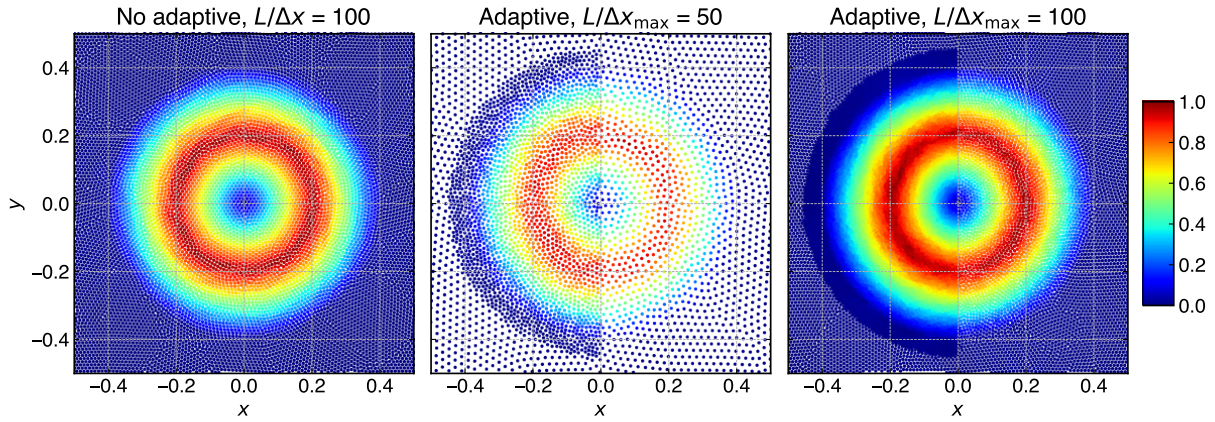


Figure 5.19: Velocity magnitude distribution for the Gresho-Chan vortex at $t = 3$ s.

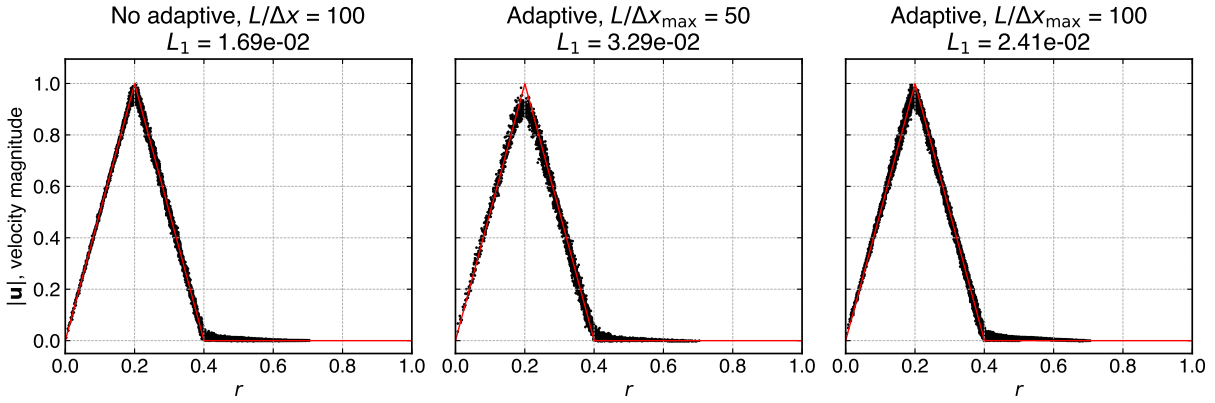
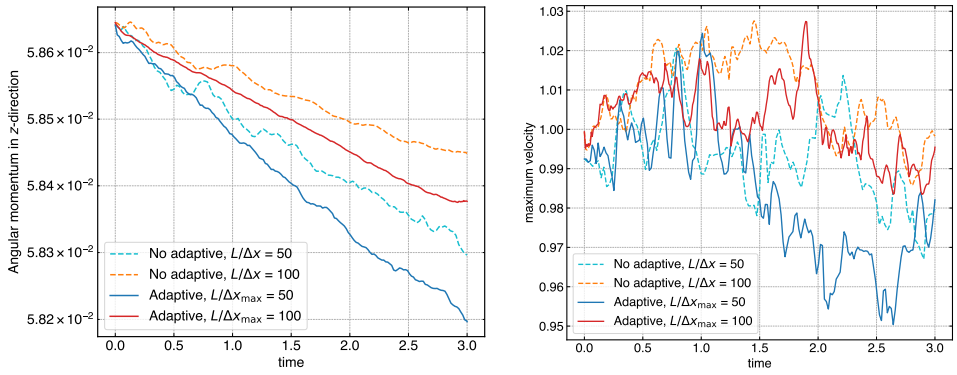


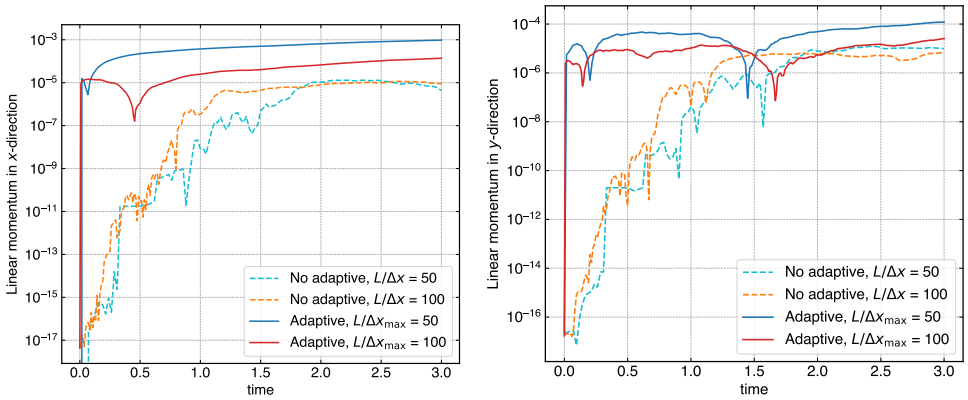
Figure 5.20: Comparison of the velocity magnitude as a function of r, the distance from the center of the vortex, to the exact solution at $t = 3$ s.

The plot shows decay and noise in the velocity magnitude. This is an inviscid problem and our simulation does not employ any artificial viscosity. It is therefore highly sensitive to small perturbations. The results show that the particle splitting and merging process introduce a small amount of noise in the simulation. However, the results show that this is only slightly dissipative. One can also see that these results are as good if not better than the existing results (Hopkins 2015; Rosswog 2015). Further, Hopkins (2015) mentions that splitting and merging can be noisy and diffusive. However, the present results show that careful splitting and merging of particles as we have done produces acceptable results. Figure 5.21a shows the angular momentum of the system as a function of time. For the non-adaptive cases we can clearly see a small amount of dissipation which reduces as we increase the resolution. A similar trend follows for the adaptive



(a) (b)

Figure 5.21: Evolution of the angular momentum (left), and the evolution of the maximum velocity (right) of the Gresho-Chan vortex.



(a) (b)

Figure 5.22: Evolution of linear momentum in x (left) and y (right) directions of the Gresho-Chan vortex.

cases. Figure 5.22 shows the evolution of linear momentum in x and y directions where higher resolution has better conservation. But due to the reasons mentioned in the remark of Section 5.3 the conservation of momentum does not hold. We note that the mass is exactly conserved in all our cases. The adaptive $L/\Delta x_{\max} = 50$ case is more dissipative compared to the $L/\Delta x_{\max} = 100$ case. To further study this, the maximum velocity evolution over time is shown in fig. 5.21b. In this figure we see that for the adaptive $L/\Delta x_{\max} = 100$ case, the maximum velocity does not decay significantly. These results affirm the accuracy of our adaptive algorithm.

5.5.3 Two-Dimensional Lid-Driven Cavity

Lid-driven cavity is a two-dimensional viscous problem with solid boundaries. We study this problem with two different Reynolds numbers 100 and 1000. We compare our results to those of Ghia et al. (1982). The domain length L is 1 m, and the top wall is moving with a velocity U_{wall} of 1 m/s. We consider a square domain $[0, L]\times[0, L]$ with two refinement levels, the intermediate refinement region $[0.3L, 0.7L]\times[0.3L, 0.85L]-[0.4L, 0.6L]\times[0.4L, 0.6L]$ consists of particles with mass twice that of the outer most region, and the inner most refinement region $[0.4L, 0.6L]\times[0.4L, 0.6L]$ consists of particles with mass four times that of the outer most and two times that of the intermediate region. We simulate with three different maximum resolutions, where $L/\Delta x_{\text{min}}$ is 50, 100 and 150. The adaptively refined regions are shown in fig. 5.23. The artificial speed of sound is $c_s = 10$ m/s, the smoothing length factor, $h/\Delta x = 1.0$, and the reference density ρ_0 is 1 kg/m^3. We non-dimensionalize the velocity by the wall velocity, $u = u^+/U_{\text{wall}}$ and $v = v^+/U_{\text{wall}}$, the pressure as $p = p^+/\rho_0 U^2_{\text{wall}}$, and the lengths by the domain length L.

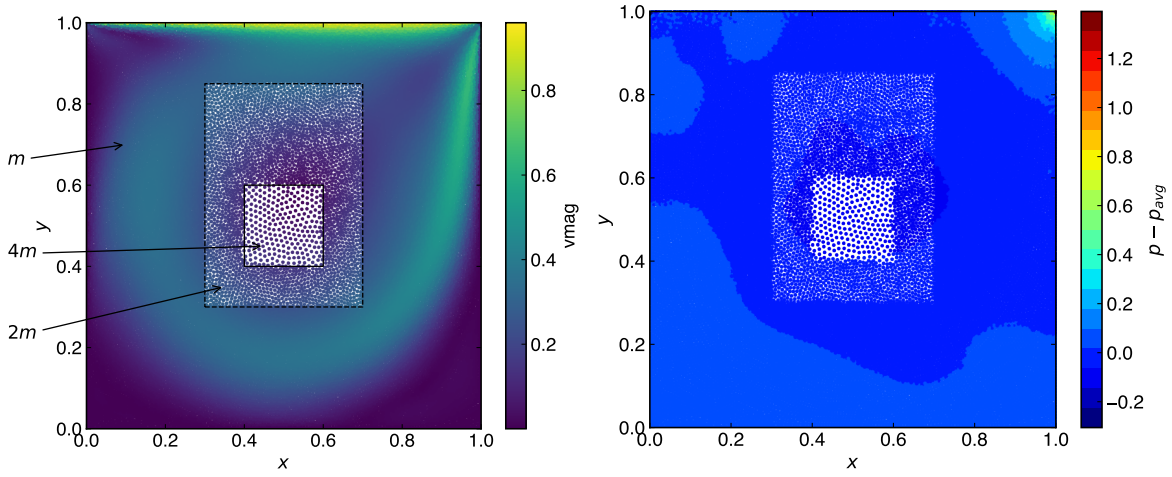


Figure 5.23: Particle plot with color indicating the velocity magnitude (left) and pressure $p-p_{avg}$ (right) for the lid-driven cavity problem simulated with $L/\Delta x_{\text{min}} = 150$ at $Re = 1000$. The regions of discretization are also indicated.

Figure 5.23 shows the velocity magnitude distribution and pressure distribution for $Re = 1000$. We use 3 layers to simulate this problem. The outer layer of particles are at the highest resolution with the particle mass corresponding to a resolution of $L/\Delta x_{\text{min}} = 150$. The middle region is at twice the mass of the outer region, this corresponds to a resolution of $L/\Delta x = 100$. The inner most resolution is the coarsest of all with an effective resolution of $L/\Delta x_{\text{max}} = 75$.

Figure 5.24 shows the centerline velocity profiles at $Re = 100$. The results match well with the results of (Ghia et al. 1982). In fig. 5.25 we show the centerline velocity profiles for $Re = 1000$. It can be observed that as the resolution is increased the centerline profiles show a good agreement with (Ghia et al. 1982).

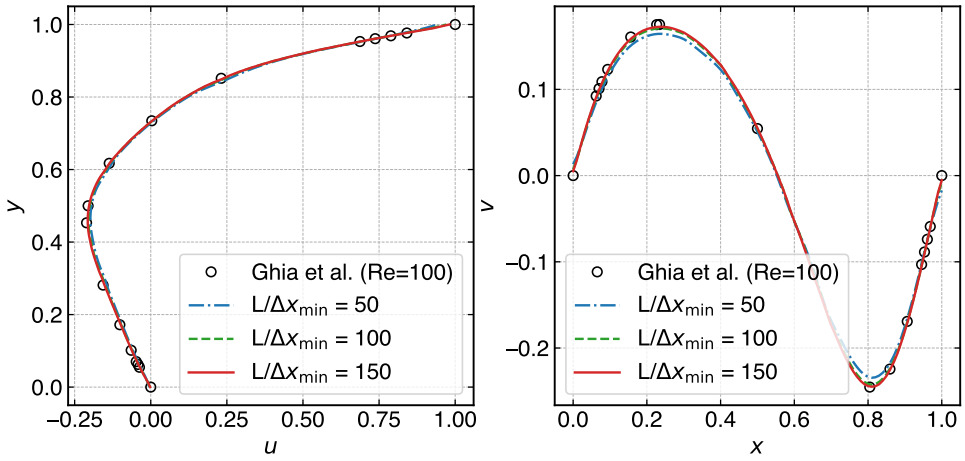


Figure 5.24: The horizontal (left) and the vertical (right) centerline velocity profiles for the lid-driven cavity at $Re = 100$ are compared with the results of Ghia et al. (1982).

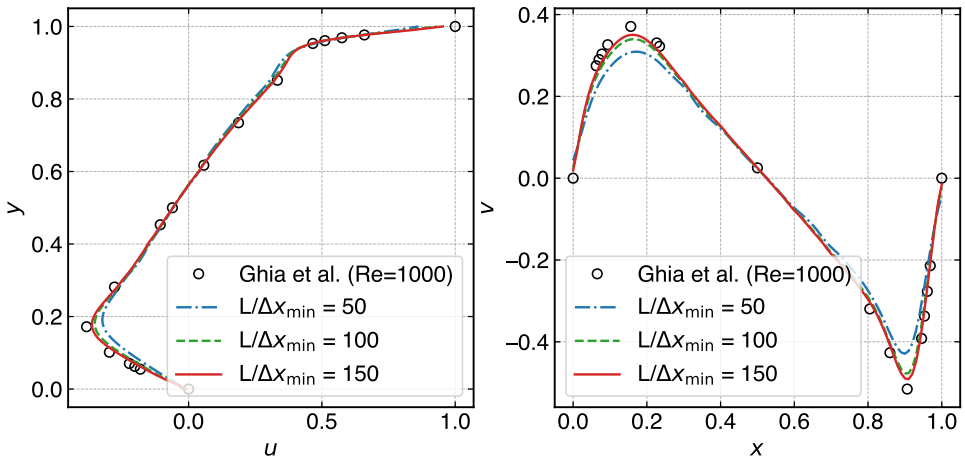


Figure 5.25: The horizontal (left) and the vertical (right) centerline velocity profiles for the lid-driven cavity at $Re = 1000$ are compared with the results of Ghia et al. (1982).

5.5.4 Flow Past a Circular Cylinder

We study the flow past a circular cylinder problem for five different Reynolds numbers ranging from 40 to 9500. We plot the coefficients of pressure drag and skin friction as a function of time. We compare the results with the high-resolution vortex method of Koumoutsakos et al. (1995) and Ramachandran (2004). Figure 5.26 shows the domain setup, where the diameter of the cylinder, which is equal to twice the radius ($D = 2R$), measures 2 meters. We use a non-dimensional time $T = tU_\infty/R$. We initialize the flow at $T = 0$ with the potential flow solution. The inlet velocity is 1 m/s, and the solid walls are inviscid. To minimize the reflection of the initial, undesirable pressure waves from the walls, we employ the non-reflection boundary conditions of Lastiwka et al. (2009) on the inviscid walls.

We simulate the problem with fixed refinement zones up to $T = 6$. For all the simulations, the coarsest resolution in the domain is $D/\Delta x_{\text{max}} = 4$. We vary the finest resolution $D/\Delta x_{\text{min}}$ from 100 to 500. For the $Re = 9500$ case we use a finest resolution $D/\Delta x_{\text{min}} = 1000$. Unless explicitly mentioned, we use a C_r value of 1.08 for all the problems. The parameters used in these simulations are summarized in table 5.3.

Table 5.3: Parameters used for the flow past a circular cylinder problem.

Quantity	Values
D, Diameter	2 m
ρ_0, reference density	1000 kg/m^3
c_s	10 m/s
$D/\Delta x_{\text{max}}$, lowest resolution	4
$D/\Delta x_{\text{min}}$, highest resolution	160, 250, 500
C_r	1.08
Reynolds number	40, 550, 1000, 3000, and 9500
Time of simulation	6
Smoothing length factor, $h/\Delta x$	1.2

To begin with, we demonstrate the advantages of using adaptive particle refinement over the non-adaptive case. We use two Reynolds numbers, 1000 and 3000. We simulate the problem up to $T = 6$ with both the non-adaptive case, using a resolution $D/\Delta x = 50$, and the adaptive case, using two different maximum resolutions $D/\Delta x_{\text{min}} = 50$ and 100. We use solution adaptivity

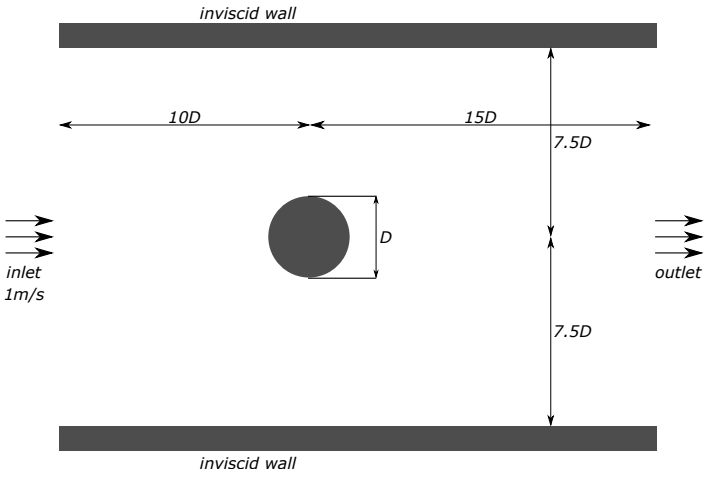


Figure 5.26: The domain dimensions for the flow past a circular cylinder problem.

based on the vorticity, and this aspect is explored in greater detail in the next section. Figure 5.27 shows the coefficients of pressure drag for the $Re = 1000$ case. The coefficients match closely for both the non-adaptive and the adaptive cases; the adaptive case with $D/\Delta x_{\min} = 100$ is slightly better. The differences between the adaptive and non-adaptive are not easy to assess in this case. On the other hand, for $Re = 3000$, the advantage of using the adaptive resolution is seen in fig. 5.28 and summarized in table 5.4. As can be seen, the adaptive and non-adaptive cases match at $D/\Delta x_{\min} = 50$. Although, at $D/\Delta x_{\min} = 50$, the adaptive case uses 35 times fewer particles and is 22 times faster than the non-adaptive case. Given the efficiency of the adaptive particle refinement, we increase the resolution in the adaptive case to $D/\Delta x_{\min} = 100$ and further to 200 and observe significantly better results in fig. 5.28. The results match those of the vortex method quite closely. For the $D/\Delta x_{\min} = 200$ case, the number of particles is 2.6 times that of the $D/\Delta x_{\min} = 50$ case with the adaptive particle refinement. This simulation requires 13 times fewer particles than the non-adaptive case at a much lower resolution of $D/\Delta x_{\min}$ of 50. These results indicate the importance and performance of the adaptive particle resolution.

Figure 5.29 shows the coefficient of skin-friction drag at different Reynolds numbers. We only show the results where the finest resolution is 500. The results are in good agreement with that of (Koumoutsakos et al. 1995; Ramachandran 2004). For the case of $Re = 9500$ the results of (Koumoutsakos et al. 1995) predicts slightly (note the logarithmic scale) higher skin-friction drag, but our results match closely to that of (Ramachandran 2004).

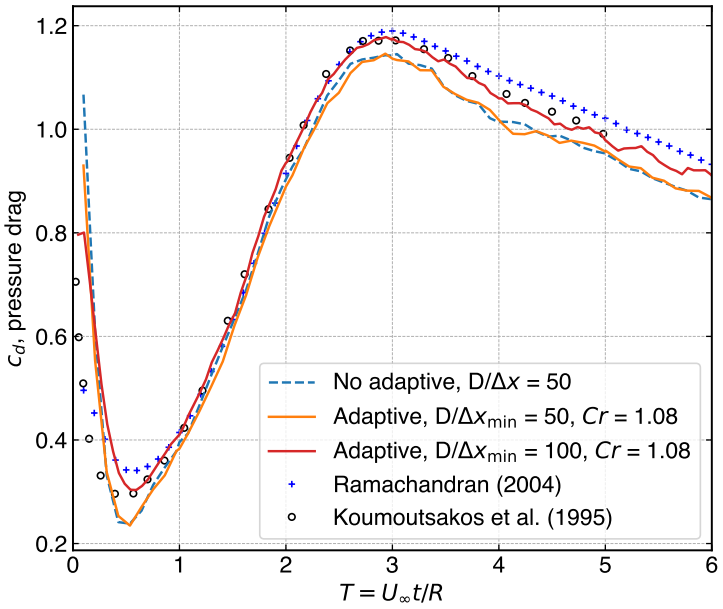


Figure 5.27: Time history of the coefficient of pressure drag for $Re = 1000$. We compare the adaptive cases with two different resolutions to the non-adaptive case with a fixed resolution of $D/\Delta x = 50$.

Table 5.4: Performance comparison of the adaptive cases with the non-adaptive cases for $Re = 3000$ at $T = 6$.

Parameter				
Adaptive	Yes	Yes	Yes	No
$D/\Delta x_{\min}$, Highest resolution	50	100	200	50
$D/\Delta x_{\max}$, Lowest resolution	4	4	4	50
time step (non-dimensional)	0.0011	0.00055	0.00027	0.0011
No. of particles	44, 091	70, 459	114, 082	1, 557, 970
CPU time taken (in mins)	8.56	27.5	96.63	192.96

In figs. 5.30 and 5.31 we plot the coefficient of pressure drag for the Reynolds numbers, 40, 500, and 1000, 3000 respectively. Our results differ from the established results at the start for up to $T = 0.5$. This is due to the weakly-compressible nature of our flow for where an initial pressure wave is required to set the velocity from the potential start to the viscous profile,

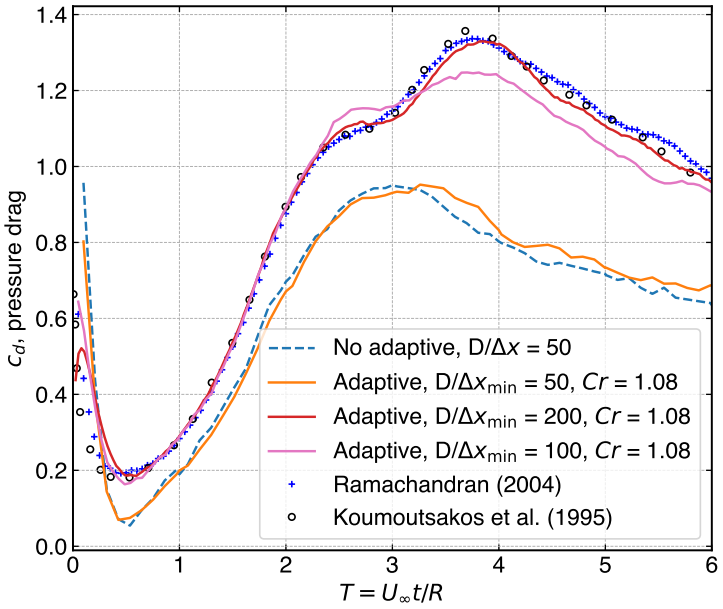


Figure 5.28: Time history of the coefficient of pressure drag for $Re = 3000$. We compare the adaptive cases with three different resolutions to the non-adaptive case with a fixed resolution of $D/\Delta x = 50$.

whereas the established results use incompressible flow. Thereafter, our results match closely with increase in the maximum-resolution.

For the $Re = 9500$ case shown in fig. 5.32 we use the finest resolution $D/\Delta x_{\text{min}} = 1000$ with a $Cr = 1.15$. The change in C_r is due to time and computational constraints. This is the highest resolution used in our simulations. Even though the characteristic features of the drag coefficient profile match with the established results the curve does not reach the maximum, the trends are consistent with the established results. We note that the resolution used by (Ramachandran 2004) corresponds to a finest resolution of $D/\Delta x_{\text{min}} = 1250$, (Koumoutsakos et al. 1995) use a million vortices for their simulation, and the present simulations employ around 200,000 fluid particles in the entire domain. We would also like to note that after three seconds maintaining symmetry is difficult and even with the DVH results of Rossi et al. (2015) there are significant differences in the drag force. The present results are clearly in good qualitative agreement with the previous works.

Figure 5.33 show the radial velocity along the axis of symmetry on the rear side of the cylinder for the Reynolds numbers 3000, with $D/\Delta x_{\text{min}} = 500$, and 9500, with $D/\Delta x_{\text{min}} = 1000$

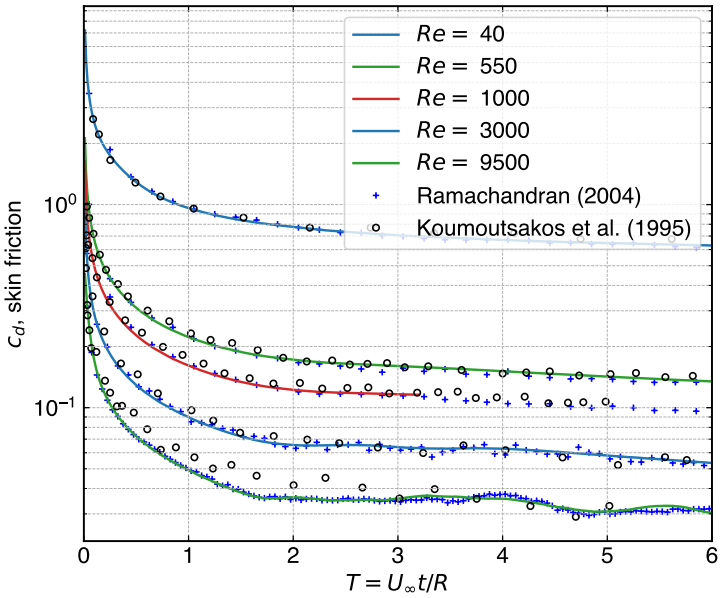


Figure 5.29: Coefficient of skin friction drag for $Re = 40$, 550, 1000, 3000, and 9500. We show only the results where the finest resolution, $D/\Delta x_{\min}$, is 500.

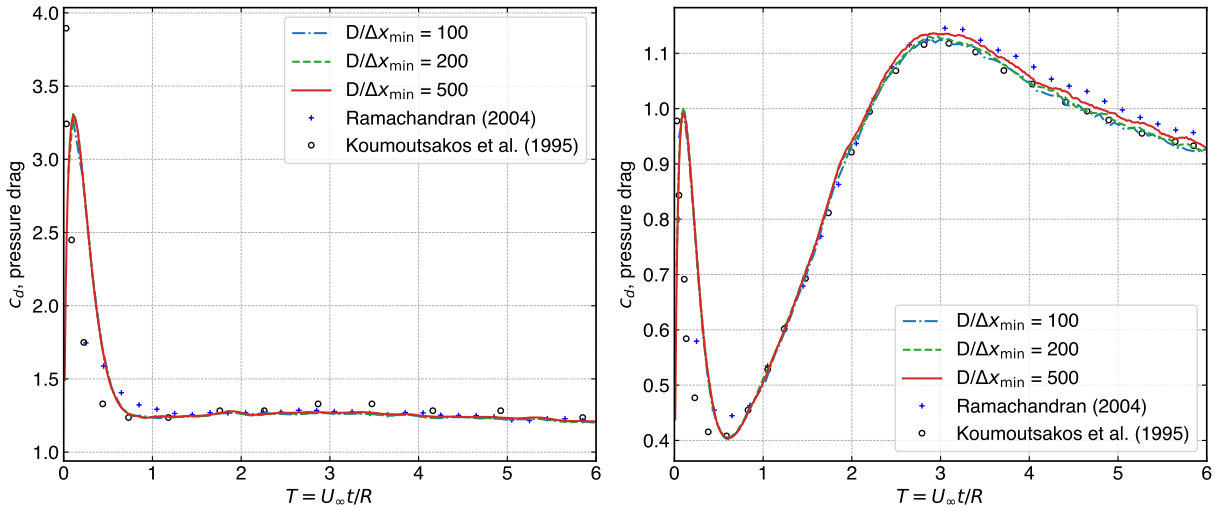


Figure 5.30: Coefficients of pressure drag at $Re = 40$ (left) and $Re = 550$ (right) as a function of time while varying the finest resolution.

and $Cr = 1.15$, at different times. In both the Reynolds numbers the results are in good agreement with Ramachandran (2004) while the results of Subramaniam (1996) show slight difference for larger time in the $Re = 3000$ case.

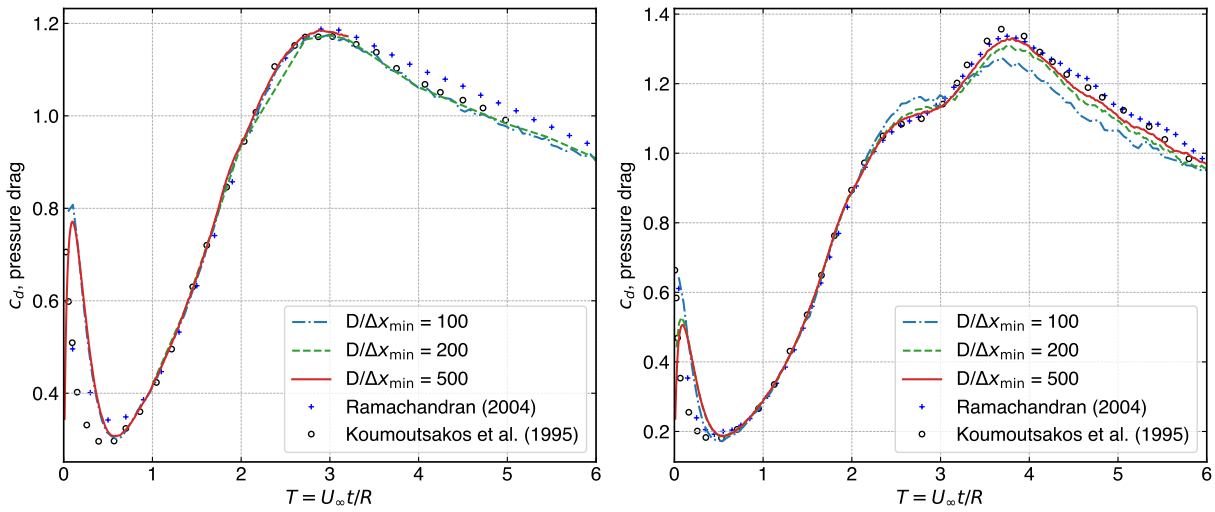


Figure 5.31: Coefficients of pressure drag at $Re = 1000$ (left) and $Re = 3000$ (right) as a function of time while varying the finest resolution.

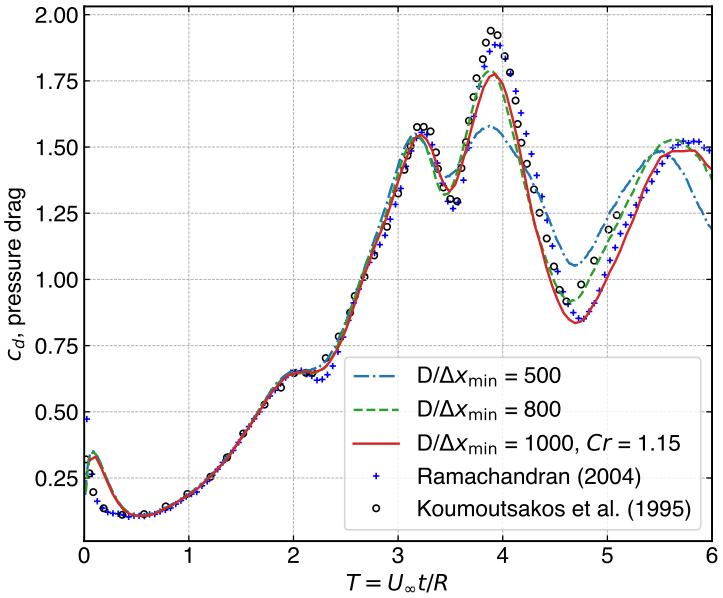


Figure 5.32: Coefficients of pressure drag at $Re = 9500$ as a function of time while varying the finest resolution.

Figure 5.34 compares the proposed formulation vorticity distribution with Durante et al. (2017). The simulation is run at the Reynolds number 1000, and times $T = 6.4$, and 12.8 are shown. In the adaptive SPH figure (left) the particles are sized proportional to the mass. There

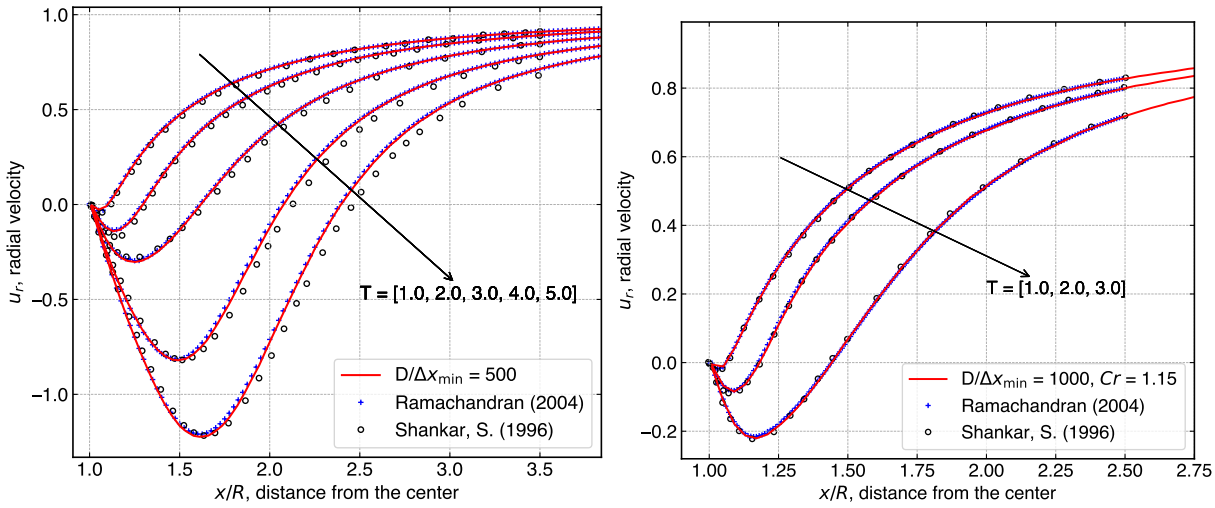


Figure 5.33: The radial velocity along the axis of symmetry in the rear of the cylinder for $Re = 3000$ (left) and $Re = 9500$ (right). The results are compared with Ramachandran (2004) and Subramaniam (1996).

are some differences in the color as a slightly different color map was used. The distribution show a good similarity.

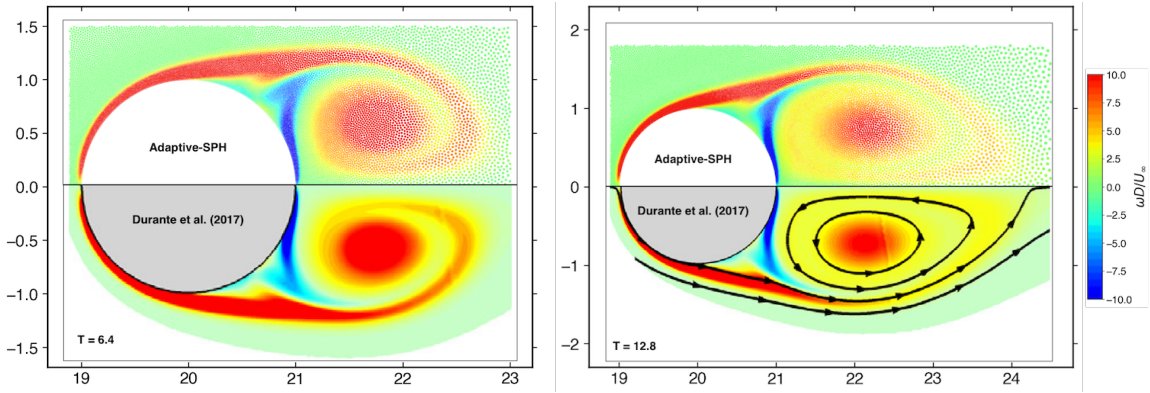


Figure 5.34: Comparison of vorticity distribution at $T = 6.4$ (left), and 12.8 (right) for the Reynolds number 1000 with the vorticity distribution of Durante et al. (2017). Adaptive-SPH particles' size is proportional to their mass. The finest resolution, $D/\Delta x_{\min}$, is 200, and the coarsest resolution, $D/\Delta x_{\max}$, is 40. The value of C_r is 1.08.

In fig. 5.35 we compare the vorticity distribution at the Reynolds number 9500 with Ramachandran (2004); times $T = 1, 2$, and 3 are shown. There are some differences in the color as

a slightly different color map was used in (Ramachandran 2004). The vortices appear to maintain the symmetry, and the secondary, tertiary, and further vortices generated at the boundary layer are captured well. The boundary layer at the leading edge of the cylinder is clearly observed. As the simulation progresses, the vortices grow big and move across different layers having different smoothing lengths since there is no solution adaptivity used in this case. At $T = 3$, the plot clearly shows the primary vortex at a different resolution than the boundary layer.

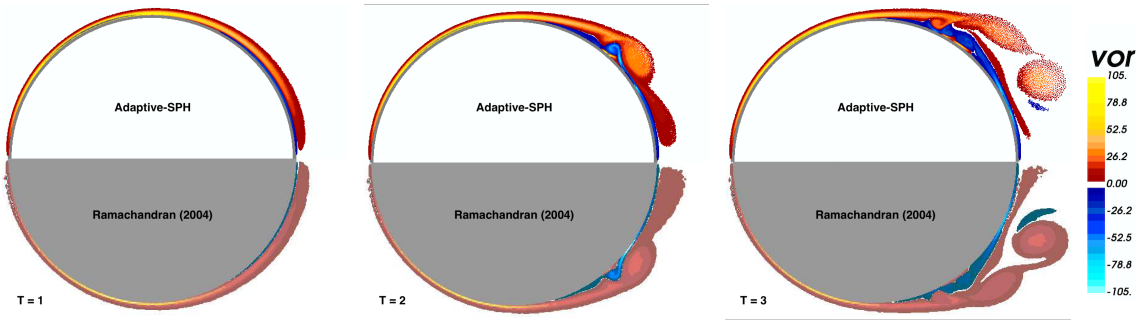


Figure 5.35: Comparison of vorticity distribution at $T = 1$ (left), 2 (center), and 3 (right) for the Reynolds number 9500 with the vorticity distribution of Ramachandran (2004). The finest resolution, $D/\Delta x_{\min}$, is 1000, and the coarsest resolution, $D/\Delta x_{\max}$, is 40. The value of C_r is 1.15.

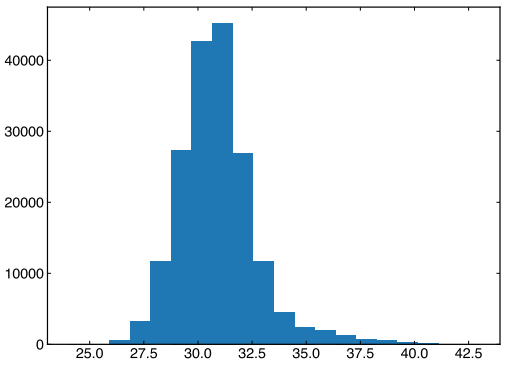


Figure 5.36: Histogram showing the number of neighbors in the simulation of $Re = 9500$ case at $T = 6$. The highest resolution, $D/\Delta x_{\min}$, of particles in this figure is 1000.

Figure 5.36 shows a histogram of the number of neighbors in the overall simulation at $T = 6$ for the resolution $D/\Delta x_{\min} = 1000$. This shows that for a majority of particles the number of neighbors are at 30. The highest number of neighbors in the simulation is at 42. This shows

the optimal neighbor distribution further maximizing the performance. In fig. 5.37 we show the smoothing length distribution for the same case. The left side shows the whole domain and the right is a zoom-in near the cylinder. The smoothing length varies across a large number of scales by a factor of 250. Even near the cylinder it varies by about a factor of 20.

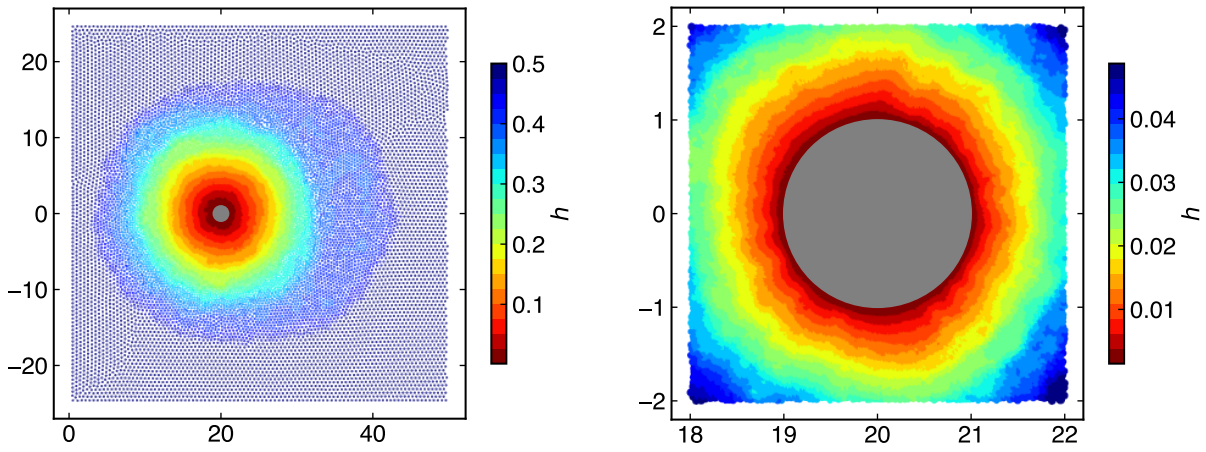


Figure 5.37: Distribution of smoothing length of each particle at $T = 6$. The highest resolution $D/\Delta x_{\text{min}}$ is 1000.

5.5.5 Solution Adaptivity: Flow Past C-Shape at $Re = 2000$

In this section we demonstrate the solution-based adaptivity. We consider a flow past C-shaped body at $Re = 2000$ and compare our results with Rossi et al. (2015), and Sun et al. (2018). The domain dimensions are given in fig. 5.38, and the smoothing length factor, $h/\Delta x = 1.2$. The outer diameter D is 1 m. Initially, we perform the simulation without solution-adaptivity and compare our results, then we show the results using solution-adaptivity. The minimum-resolution $D/\Delta x_{\text{min}}$ is 25 and the maximum-resolution $D/\Delta x_{\text{min}}$ is 200. The minimum and maximum-resolution match the respective resolutions of Sun et al. (2018). We simulate the problem for 30 seconds.

In fig. 5.39 we show the coefficients of total drag and lift. The results are in good agreement with Rossi et al. (2015). The initial noise within $T = 1$ is due to the weakly-compressible nature of our formulation. We compare the number of particles used in our simulation with the simulation of Sun et al. (2018). Sun et al. (2018) does not mention the total length of the domain instead provides the dimensions $[2.75, 5.75]\times[-0.8, 1.2]$ of an inner rectangular domain

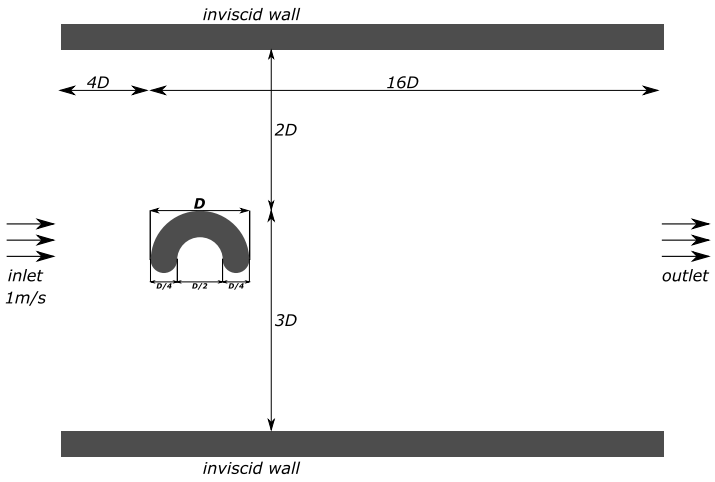


Figure 5.38: The domain dimensions for the flow past C-shape problem.

containing the C-shape body, where the minimum-resolution is 50. We estimate the number of particles inside this domain to be approximately 98,000, whereas for our simulation the number of particles in this domain is about 38,735. This shows that we use 2.53 times lower number of particles and achieve significantly better results. This further demonstrates the efficiency and the accuracy of our method.

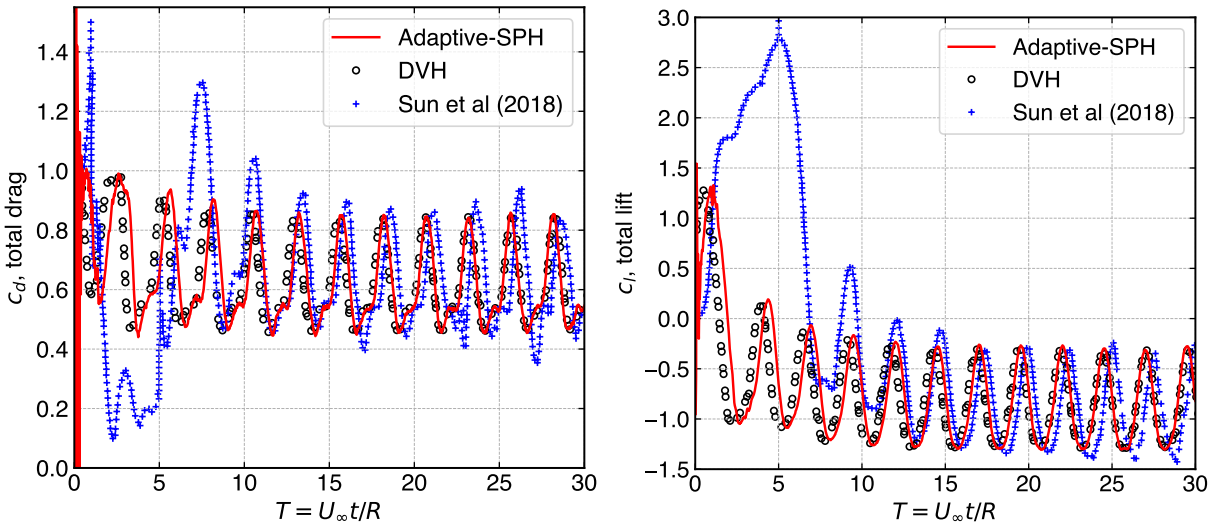


Figure 5.39: The coefficient of drag (left) and lift (right) for the flow past C-shape simulation at $Re = 2000$. The results are compared with Rossi et al. (2015) and Sun et al. (2018).

Now, we simulate the flow past C-shape using solution-adaptivity, where the vorticity in the flow is monitored and particles with absolute vorticity value above 5% of the maximum vorticity are resolved to the highest resolution. In this simulation we use the maximum-resolution $D/\Delta x_{\min}$ of 125. The simulation is performed for $T = 20$. In fig. 5.40 the coefficients of total drag and lift are shown which are in good agreement with the results of (Rossi et al. 2015).

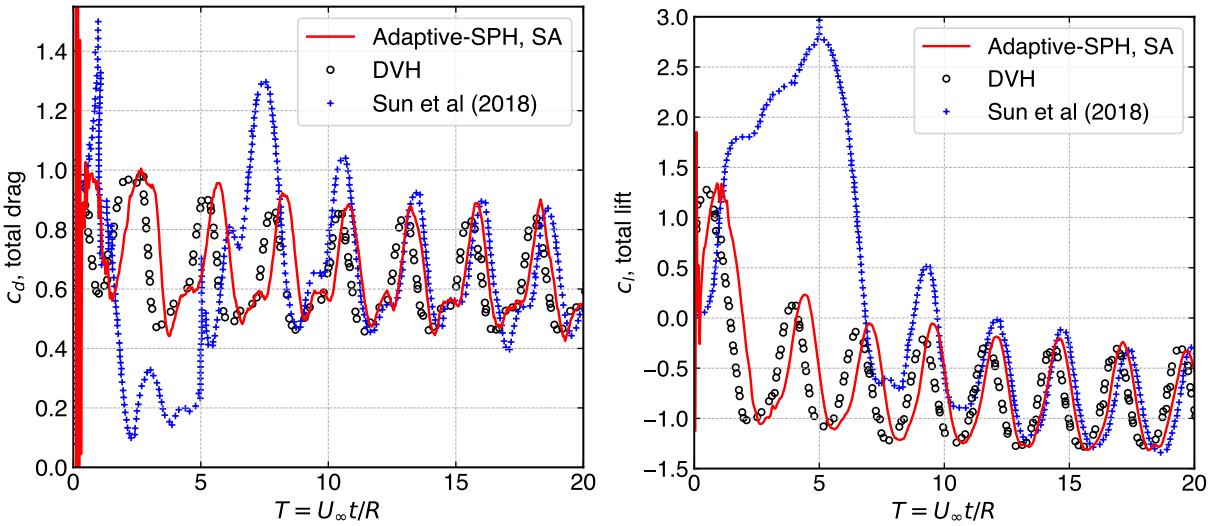


Figure 5.40: The coefficient of drag (left) and lift (right) for the flow past C-shape simulation at $Re = 2000$ with solution adaptivity are compared with Rossi et al. (2015) and Sun et al. (2018).

Figure 5.41 shows the vorticity distribution of the particles for the non-solution adaptive and the solution adaptive cases at $T = 20$. It can be seen that for the non-solution adaptivity case the trailing vortices are not refined after they move certain distance away from the C-shape body, whereas in the solution-based adaptivity the trailing vortices are resolved to the highest resolution, based on the cut-off criteria stated above. In fig. 5.42a we show the zoomed-in view near the C-shape body for the solution-based adaptive case, and in fig. 5.42b we show the smoothing length distribution demonstrating the effect of solution adaptivity.

The results of this section demonstrate the accuracy and efficiency of the proposed method even when there is a large change in the resolution. The method shows little dissipation and it is capable of performing a high-resolution simulation of a $Re = 9500$ flow. The results show good accuracy with the solution-based adaptivity.

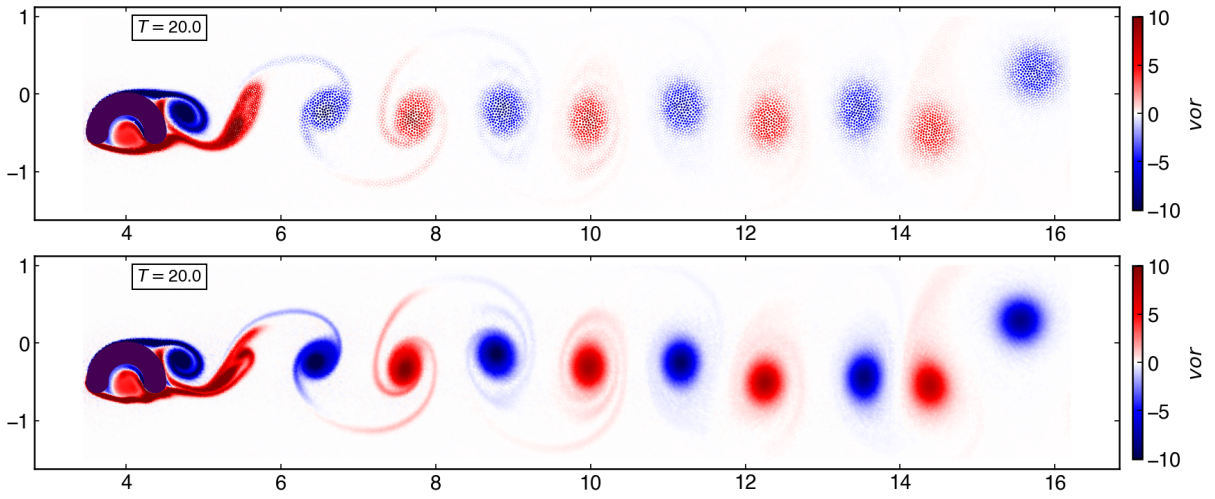
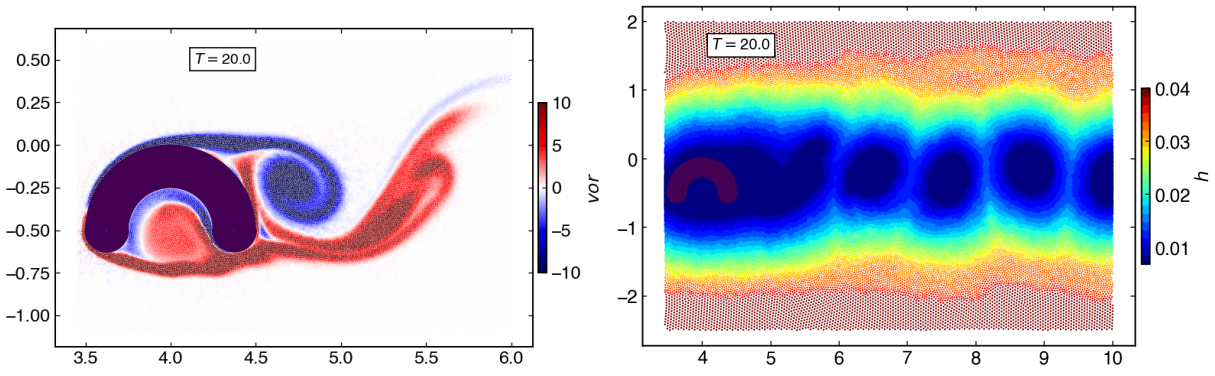


Figure 5.41: Vorticity distribution around the C-shape body simulated using adaptive-SPH, without solution-adaptivity (top) and with solution-adaptivity (bottom) at $T = 20$, respectively.



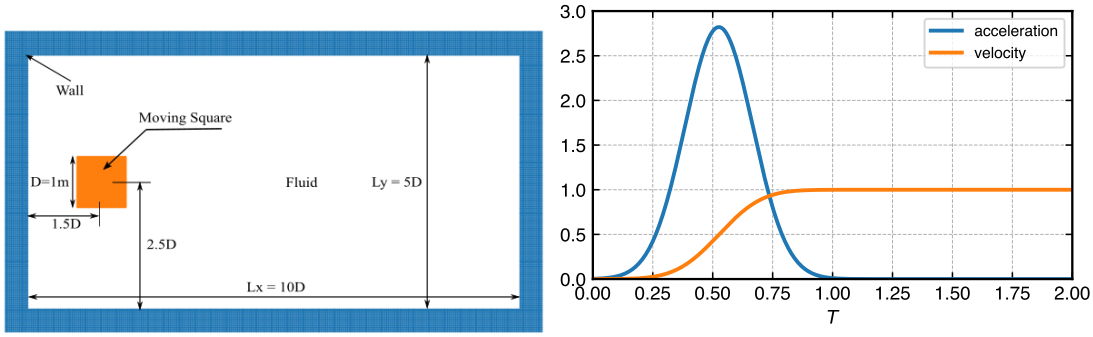
(a) (b)

Figure 5.42: (a) Zoomed-in view of the vorticity, (b) smoothing length distribution around the C-shape body simulated using the adaptive-SPH, with solution-adaptivity at $T = 20$.

5.5.6 Flow Around a Moving Square

The flow around a moving square represents a simple geometry but a flow situation that can be very complex due to the sharp edges. The problem consists of a square geometry moving through a stationary fluid kept inside a rectangular box. The test can generate intense unsteady vorticity and has been a challenging test case for SPH solvers. This simulation is part of a validation test suite compiled by the SPH rEsearch and engineeRing International Community

(SPHERIC) (*SPH rEsearch and engineeRing International Community* 2022). The benchmark results of this test case are simulated using Finite Difference Navier-Stoked (FDNS) solver (Colicchio et al. 2006). In SPH, Vacondio et al. (2013a) and Marrone et al. (2013b) studied this problem. We study the time histories for the drag force coefficients (pressure, viscous or total) and the generation and diffusion of vorticity.



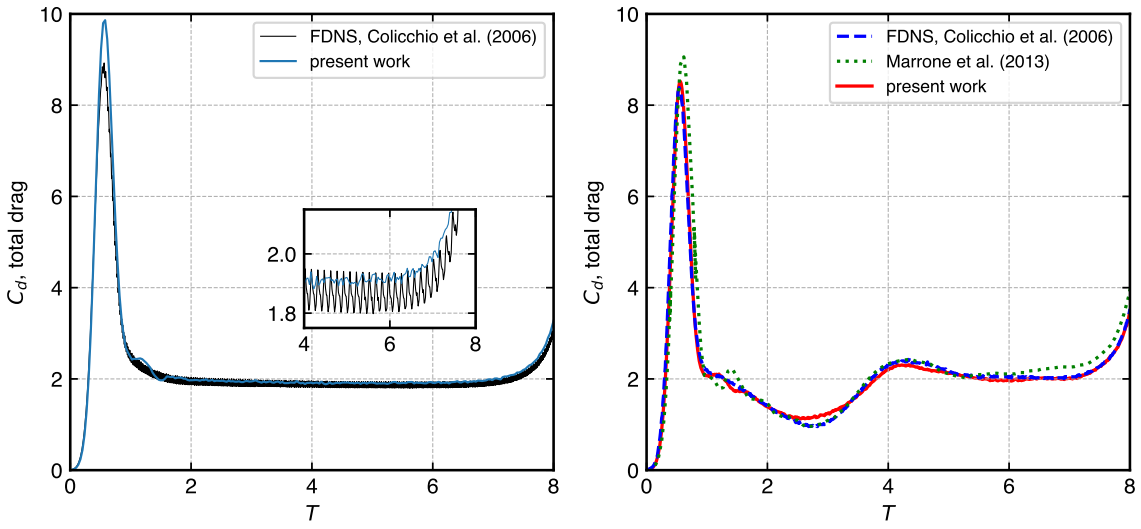
(a) (b)

Figure 5.43: Flow around a moving square, (a) Moving square test case geometry, (b) the body motion in time.

The geometric configuration is shown in fig. 5.43a at time $t = 0$. The body motion in time is shown in fig. 5.43b. The domain size is $L_y \times L_x = 5\text{ m} \times 10\text{ m}$, with the square obstacle having a length of 1m. The square motion is prescribed with a smooth acceleration in the x-axis starting from rest until it reaches a steady maximum velocity $U_{\text{square}} = 1.0$ m/s. The boundary conditions are no-slip and no-reflection on all walls of the rectangular domain and slip velocity on the solid square. The simulation is conducted for a Reynolds number of 100 with no gravity force and zero initial pressure field for the viscous incompressible Newtonian fluid of density $1\,\text{kg m}^{-3}$.

In fig. 5.44, we show the time evolution of the drag coefficient of a moving square at (a) $Re = 100$ using $D/\Delta x_{\text{min}} = 80$ and (b) at a slightly higher Re of 600 using $D/\Delta x_{\text{min}} = 200$. Simulated results obtained from the present method are compared with the results of the FDNS solver (Colicchio et al. 2006) and Marrone et al. (2013b). The comparison shows, especially with FDNS, very good agreement with all the necessary features.

The simulation requires $44k$ fluid particles using adaptive resolution for the converged solution. The initial spacing of particles on the domain is $\Delta x_{\text{max}} = 0.04$; the smallest spacing is $\Delta x_{\text{min}} = 0.0125$ near the obstacle. The solid particles are packed using the minimum resolution to create a mapping in the fluid-solid interface. If the Δx_{min} was used in the entire domain,



(a) (b)

Figure 5.44: Comparison of the total drag coefficient for a square cylinder at (a) $Re = 100$ and (b) $Re = 600$. Simulated results of adaptive EDAC-SPH are compared with FDNS solver (Colicchio et al. 2006) and Marrone et al. (2013b).

$D/\Delta x_{\min}$ would be 80 which produces $320k$ as the total number of particles. However, with an adaptive particle resolution, we have used only $44k$ particles to produce accurate results. This substantially reduces the number of particles by 7.273 times when compared with Marrone et al. (2013b) and by four times when compared with the results by (Colicchio et al. 2006).

Figure 5.45 shows the variation of drag coefficient for flow around a moving square at different Reynolds numbers. The comparison is made for three Reynolds numbers of 50, 100, and 150. The results of the adaptive EDAC-SPH are in good agreement with (Colicchio et al. 2006) results. However, due to numerical reasons (Colicchio et al. 2006) show the small amplitude and high-frequency oscillations which are avoided in our case.

Figure 5.46 show the particle distribution with color indicating pressure, vorticity and velocity magnitude taken at $T = 5$ and $T = 8$. The qualitative observations illustrate the generation, diffusion, and convection of the unsteady vorticity proving the capability of the adaptive EDAC-SPH scheme.

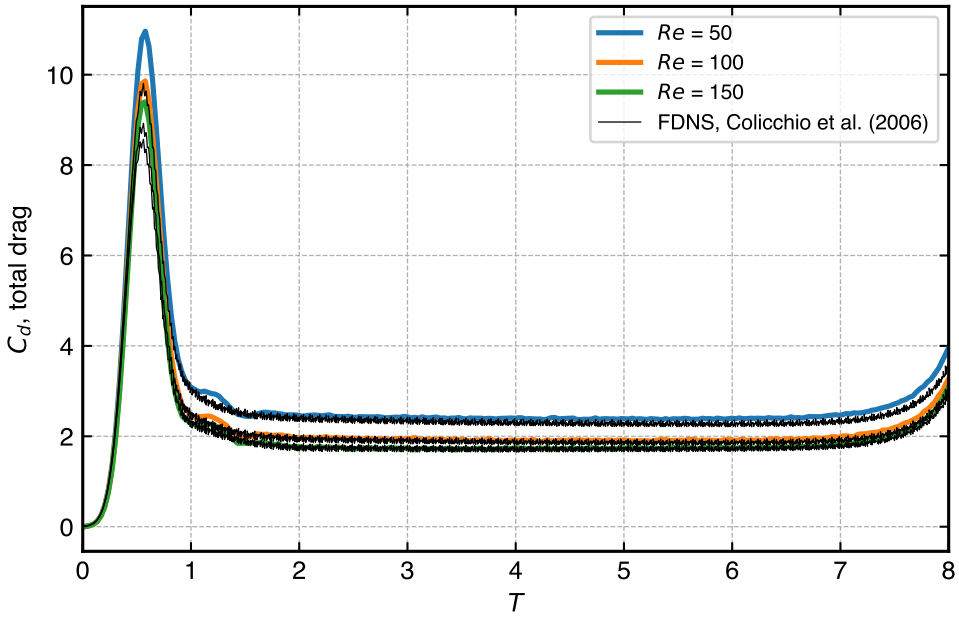


Figure 5.45: Comparison of results against the results of Colicchio et al. (2006), with varying Reynolds number, for flow around a moving square.

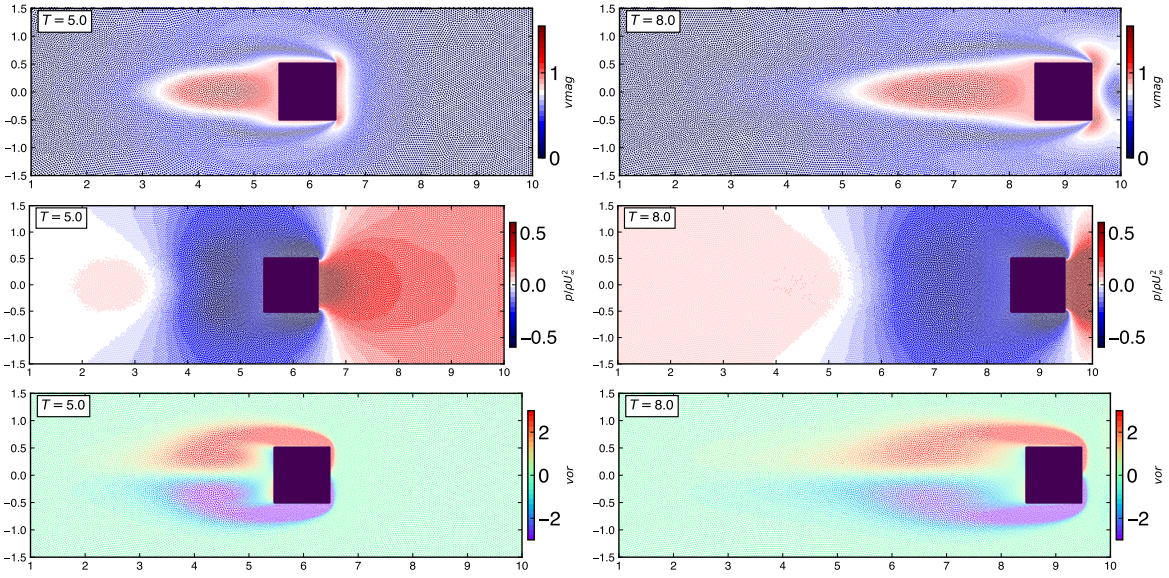
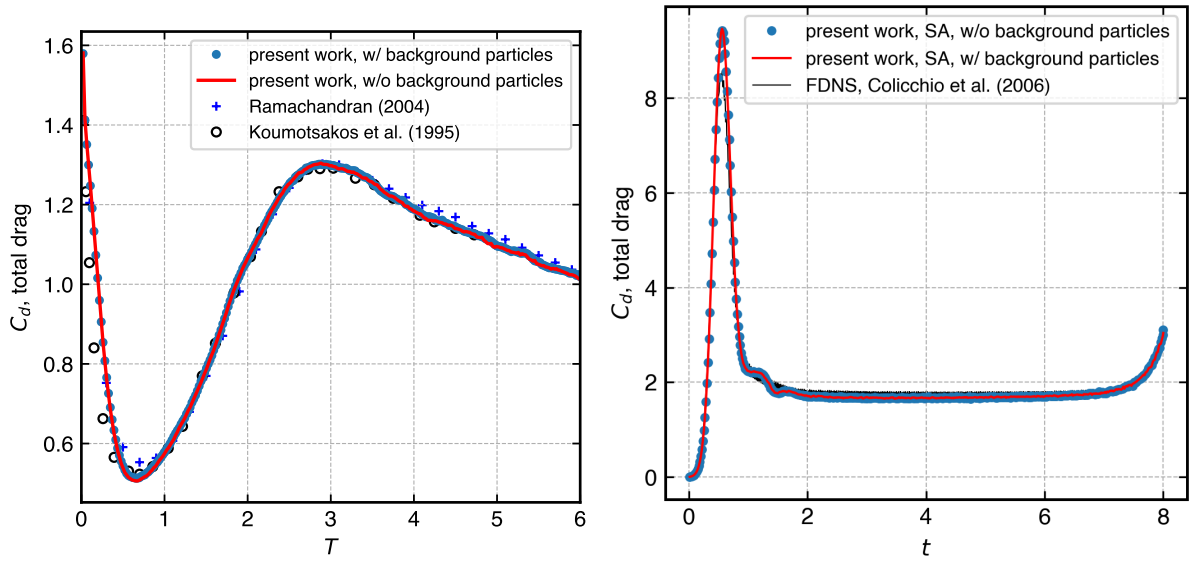


Figure 5.46: Particle distribution of moving square problem at $T = 5$ (left), and $T = 8$ (right) for $Re = 100$ showing: velocity magnitude (top), pressure (middle), and vorticity (bottom). The lowest resolution used is $D/\Delta x_{\max} = 25$ and the highest resolution used is $D/\Delta x_{\min} = 80$.

5.5.7 Removal of Background Particles

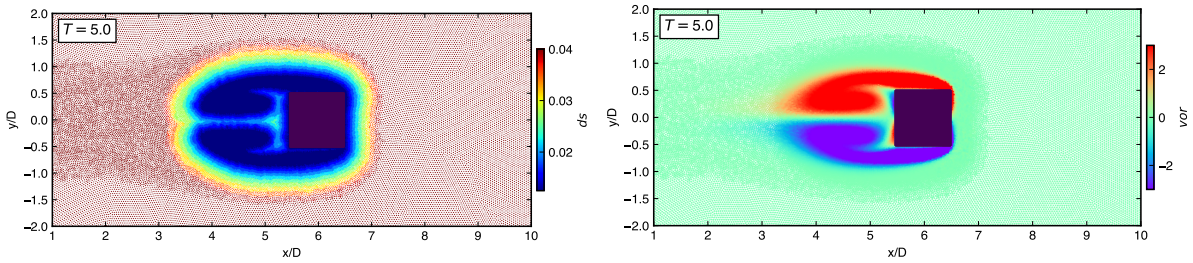
In Section 5.4.3, we used a collection of particles called "background" particles to define the particle spacing Δs and reference mass. But here, we show that the background particles are not necessary to set this reference mass. Updating the reference mass on the particles themselves (see, algorithm 6) produces results closely matching those produced by the original algorithm using background particles. This reduces memory usage, and also provides a small reduction in the computational time. We perform two simulations to establish the no background particle approach. We start by, considering the flow past a circular cylinder at $Re = 1000$, the simulation parameters and domain sizes are the same as defined in Section 5.5.4. Furthermore, we compare the results with the established vortex method results (Koumoutsakos et al. 1995; Ramachandran 2004). We then consider the flow past a moving square in a box at $Re = 150$. Refer to Section 5.5.6 for the simulation parameters. We compare the results to the incompressible FDNS simulation results (Colicchio et al. 2006). In this case, the geometry is moving. In addition, we use solution adaptivity to refine the particles to the lowest resolution where the vorticity exceeds 5% of the maximum vorticity in the simulation.

Figure 5.47a shows the time history of the drag coefficient of the flow past cylinder problem without solution adaptivity. The results of adaptive EDAC-SPH are computed with and without use of background particles, and both results match very well with the reference data from vortex method simulations. Figure 5.47b shows the time history of the drag coefficient of the moving square problem using solution adaptivity. This further confirms a good match when not using the background particles. Figure 5.48a shows the distribution of the spacing Δs without using background particles and in fig. 5.48b we show the vorticity distribution. It can be seen that the regions satisfying the solution adaptivity criteria are refined to the highest resolution, and the particles closest to the solid square particles are refined to the highest level.



(a) (b)

Figure 5.47: (a) Comparison of the drag coefficient vs time with and without the use of background particles to that of the vortex method results(Koumoutsakos et al. 1995; Ramachandran 2004). (b) Time history of the drag coefficient of a moving square simulated using solution adaptivity (SA) at $Re = 150$ compared against FDNS results(Colicchio et al. 2006).



(a) (b)

Figure 5.48: (a) Δs distribution (b) Vorticity distribution for the flow past a moving square at $Re = 150$, the highest resolution $D/\Delta x_{\min}$ is 80 and the lowest resolution $D/\Delta x_{\max}$ is 25, and a C_r spacing ratio of 1.12 is used.

5.5.8 Performance Analysis

In this section, we discuss the parallel performance of the adaptive EDAC-SPH implemented using the framework of PySPH. To show that our adaptive algorithm is fully parallel, we run the moving square test case with the highest resolution of $\Delta x_{\text{min}} = 0.0075$ and the lowest resolution of $\Delta x_{\text{max}} = 0.2$, giving us 57k particles. We also do not employ background particles. As opposed to the gradual start in the moving square problem described in Section 5.5.6 we move the solid with a uniform velocity of 1 ms^{-1}. We proceed to test this problem on a 6-Core Intel Core i7 CPU. We compare the time taken to run 1000 time-steps on a serial single-core single-thread execution with a parallel (OpenMP) execution on the multi-core CPU with 4-threads and 6-threads.

Table 5.5: The time break-up for running 1000 time-steps of the moving square problem executed on a serial single-thread, and parallel 4-threads and 6-threads on a 6-core CPU.

Component	1-thread	4-threads	6-threads
splitting & merging	24.74 s	7.91 s	6.64 s
removing & adding particles	0.63 s	0.61 s	0.70 s
EDAC scheme	260.77 s	78.44 s	64.38 s

Table 5.5 shows the time taken by various algorithms described in this work along with the time taken by the adaptive EDAC-SPH scheme. The splitting and merging component constitutes the Update Spacing algorithm 6, splitting algorithm 4, and merging algorithm 5; the removing and adding of particles constitutes for the time taken by the removal of the merged particles' indices from the data structure, and addition of new particles to the data structure; the EDAC scheme constitutes the overall time taken by the SPH formulation.

Table 5.5 shows that the multi-thread implementation scales well in parallel with a scale-up of 3.13x on 4-threads and 3.73x on 6-threads. The time taken for removing and adding particles to the data structure constitutes less than 2.5% of the adaptive algorithm and this process is serial in execution. The present scheme again scales well with an increase in number of threads. The scale-up is 3.32x on 4-threads and 4.05x on 6-threads. The parallel execution of the adaptive algorithm constitutes 10 to 11% of the adaptive EDAC-SPH scheme.

The results of this section validate our formulation with the existing numerical simulations.

5.5.9 Complex Motion Demonstration: Rotating S-Shape

Simulation of fluid flow around moving solid bodies is challenging and even more challenging in complex motion scenarios such as the rotation of a complex geometry and motions that combine pitching and translating. The adopted adaptive EDAC-SPH method automatically refines the particles around solid bodies performing any type of motion. Applications in complex moving cases are demonstrated in this section. We consider a rotation of an S-shape geometry for the demonstration. All the test cases use boundary conditions of the moving square problem discussed in Section 5.5.6 for a stationary fluid. We consider a counter-clockwise rotating S-

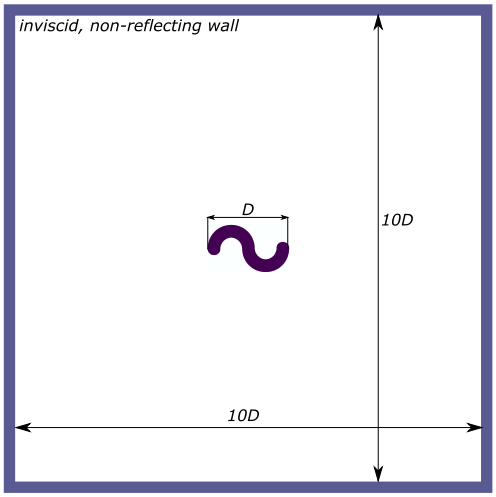


Figure 5.49: Schematic diagram of the rotating S-shape, where the outer diameter D of the S-shape is 2 m.

shaped body at $Re = 2000$. The outer diameter D is 2 m and fig. 5.49 shows the domain schematic. The minimum-resolution $D/\Delta x_{\max}$ is 10 and the maximum-resolution $D/\Delta x_{\min}$ is 200. The body is rotated at a frequency of 0.1 sec^{-1}. We simulate the problem for $t = 10$ sec. Figure 5.50 shows the vorticity distribution of the particles for the rotating S-shape at $t = 2, 5, 8, 10$ secs. No background particles are used in this simulation.

This section demonstrates the capabilities of the adaptive particle refinement in simulating complex motion and multiple bodies. This motivates further the use of adaptive EDAC-SPH in a range of applications.

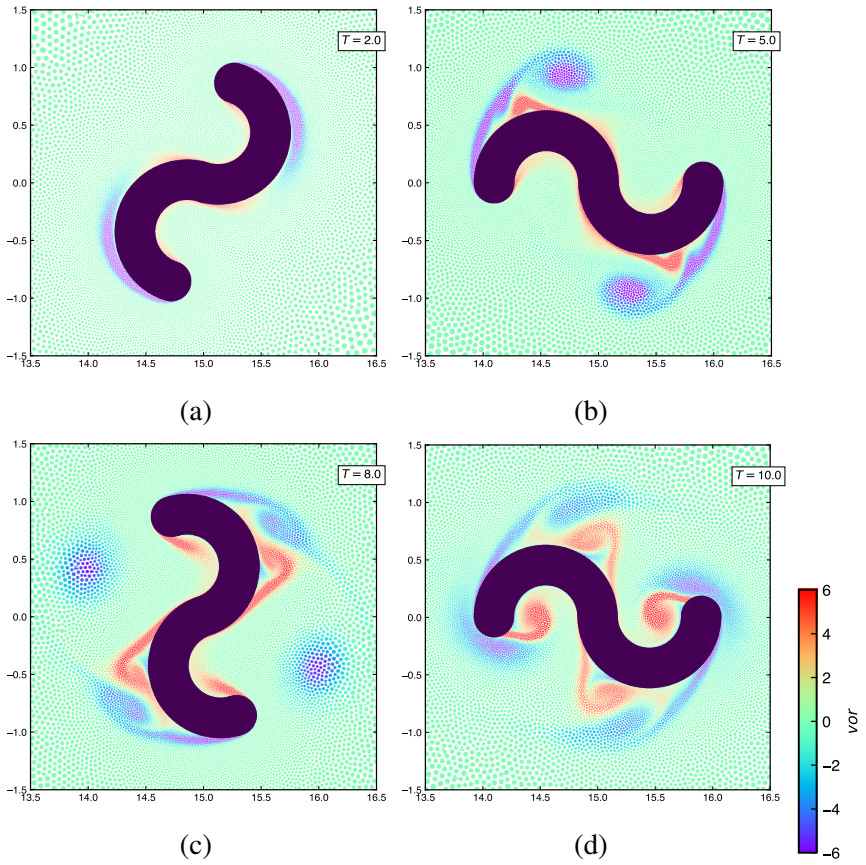


Figure 5.50: Vorticity distribution at different times around the counter-clockwise rotating S-shape at $Re = 2000$. The point size is proportional to the mass of the particle.

5.6 Summary

We have proposed an accurate and efficient method to handle adaptive resolution in the context of weakly-compressible SPH. This is achieved using (i) an accurate EDAC scheme (Ramachandran et al. 2019) along with the recent corrections of (Adepu et al. 2021), the use of variable-h corrections of (Vacondio et al. 2012b), and particle shifting (Lind et al. 2012); (ii) adaptive splitting and merging of particles where care is taken to ensure that the number of particles is minimum and the number of neighbors is optimal. We employ background particles to specify the regions of refinement. Importantly, the method allows for specifying fixed regions of refinement, automatic geometry-based refinement, and automatic solution-based adaptivity elegantly in the same framework. The algorithms employed are parallel. We provide an open-source im-

plementation of the entire algorithm along with complete automation of all the results presented in this work.

We demonstrate the accuracy of the method using several benchmarks. The Taylor-Green vortex and Gresho-Chan vortex benchmark problems clearly demonstrate that the method is not diffusive and is more accurate than other recent adaptive refinement techniques.

6 Adaptive Dual-Time Smoothed Particle Hydrodynamics

Every now and then a paper appears with a title like, 'A method to solve all partial differential equations.' The content of such papers is always very far from satisfying the claims made in the title. It is rumoured that a paper of this kind inspired Lewy to construct his famous example of a linear PDE which has no solutions at all.

—— Joel Smoller, *Shock waves and reaction-diffusion equations (1994)*

6.1 Introduction

Now comes the time to integrate the fast and the adaptive methods we formulated in the preceding chapters. Here, we are going to use DTSPH (chapter 4), with variable-h corrections, as the scheme of choice. The choice of choosing DTSPH over SISPH comes down to the ease of implementation of the method. The adaptive resolution algorithm does not change at all, and the execution of this algorithm comes before the DTSPH scheme in the computational process. We do not use background particles in the simulations done in this chapter. In the next section, we describe the Adaptive-DTSPH formulation. The source code of this chapter which runs the test cases and reproduces the figures is provided at https://gitlab.com/pypr/adaptive_dtsph.

6.2 Formulation

We are not moving the particles in pseudo-time so we employ equations from Section 4.3.2. The EDAC pressure equation in the variable-h formulation is given by,

$$\begin{aligned}\frac{\partial p_i}{\partial \tau} = \frac{\rho_0 c_s^2}{\beta_i} \sum_{j\in N(i)} \frac{m_j}{\rho_j} \mathbf{u}_{ij} \cdot \nabla W(r_{ij}, h_i) \\ + \frac{1}{\beta_i} \sum_{j\in N(i)} \frac{m_j}{\rho_j} \nu_{\mathrm{e},ij} \frac{p_{ij}}{(r_{ij}^2 + \eta \mathbf{h}_{ij}^2)} \nabla W(r_{ij}, h_i) \cdot \mathbf{x}_{ij},\end{aligned} \tag{6.1}$$

where, $\nu_{\mathrm{e},ij}$ is same as given in the previous chapter, eq. (5.31) with a values of 1.5 for α_e used in all our simulations. The momentum equation in the variable-h form is given by,

$$\begin{aligned}\frac{\partial \mathbf{u}_i}{\partial \tau} + \frac{d\mathbf{u}_i}{dt} = - \sum_{j\in N(i)} m_j (P_i \nabla W(r_{ij}, h_i) + P_j \nabla W(r_{ij}, h_j)) \\ + \frac{1}{\beta_i} \sum_{j\in N(i)} m_j \frac{4\nu \nabla W_{ij} \cdot \mathbf{r}_{ij}}{(\rho_i + \rho_j)(r_{ij}^2 + \eta h_{ij}^2)} \mathbf{u}_{ij}.\end{aligned} \tag{6.2}$$

where, P_i and P_j are given by eq. (5.27). The complete algorithm is given in algorithm 7. Since, the adaptive algorithm comes before the scheme algorithm in the program execution order, the scheme proposed in this chapter can be easily replaced with the EDAC scheme formulated in the previous chapter.

Algorithm 7 Fixed particles in pseudo-time, see Section 4.3.2.

while $t < t_{\text{final}}$ **do**
  for all particles **do**
    compute h using eq. (5.54)
    compute ρ using eq. (5.25)
    compute $\left(\frac{\partial p_i}{\partial \tau}\right)^k$ using eq. (6.1)
    compute $\left(\frac{\partial \mathbf{u}_i}{\partial \tau}\right)^k$ using eq. (6.2)
  for all particles **do**
    predict velocity using eq. (4.10)
    predict position using eq. (4.11)
  while check convergence using eq. (4.39) **do**
    for all particles **do**
      compute $\left(\frac{\partial p_i}{\partial \tau}\right)^{k+\frac{1}{2}}$ using eq. (6.1)
      update $\left(\frac{\partial \mathbf{u}_i}{\partial \tau}\right)^{k+\frac{1}{2}}$ using eq. (6.2)
    for all particles **do**
      update to $\mathbf{v}^{k+1}$, and $\mathbf{p}^{k+1}$ using eqs. (4.21) and (4.23)
  update positions to $\mathbf{r}^{n+1}$ using eq. (4.26)
  shift particles using eq. (2.26)
  correct the velocities and pressure using eq. (2.28)

6.3 Results and Discussions

Numerical simulations to show evidence for the performance benefits of using a fast scheme with adaptivity are shown in this section. We re-visit the flow past circular cylinder at *Re* 9500 and the flow past C-shape problems at *Re* 2000. The reason for this is that these problems form the starting point after which more complex problems are simulated.

6.3.1 Flow Past a Circular Cylinder

The computational setup and the initial conditions are same as the one we have previously used in Section 5.5.4. We use a tolerance of 10^{-3} for convergence in the iterations of DTSPH

scheme. Coefficient of drag profile is shown in fig. 6.1 as the highest resolution is varied from $D/\Delta x_{\min} = 200$ to 1000. The results are compared with the existing vortex method results from the literature. As the resolution is improved, the trend matches within a reasonable range of the benchmark results.

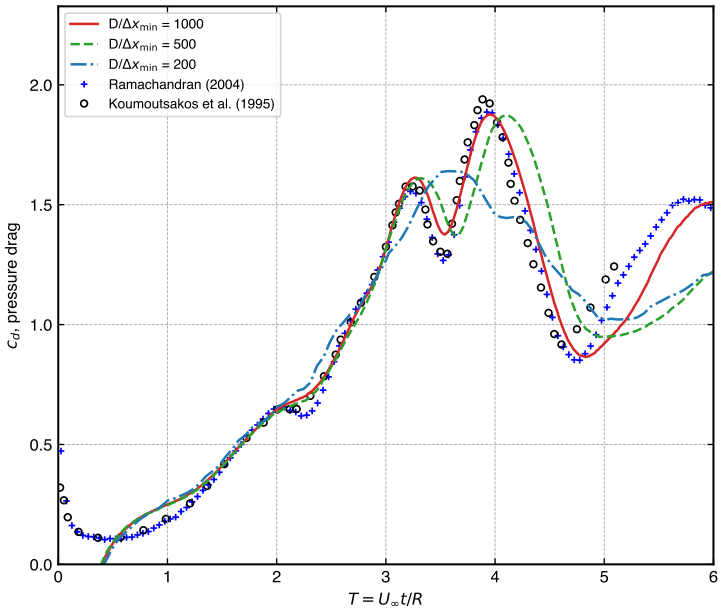


Figure 6.1: Flow past cylinder at $Re = 9500$ simulated using Adaptive-DTSPH. Shown are the coefficients of pressure drag as a function of time while varying the finest resolution.

Figure 6.2 shows the coefficients of drag comparison of Adaptive-DTSPH with Adaptive-EDAC scheme. We only show the simulation where the highest resolution, $D/\Delta x_{\min}$ is 1000. In this case we use 300k particles in the full domain where the lowest resolution is $D/\Delta x_{\max} = 40$ and resolution varies with a Cr of 1.08 giving a mass ratio of 1:250. The Adaptive-EDAC scheme has the same maximum resolution but a Cr of 1.15 giving 200k fluid particles. The plot shows the results are within acceptable range; Adaptive-DTSPH captures the peaks well; and the vortex shedding, which indicates the drop in drag is also in phase with the existing results. The plot on the right hand side shows the skin-friction drag that matches well with Adaptive-EDAC. Figure 6.3 shows the vorticity distribution for both the cases, note that the particles are scaled proportional to their mass in the visualization.

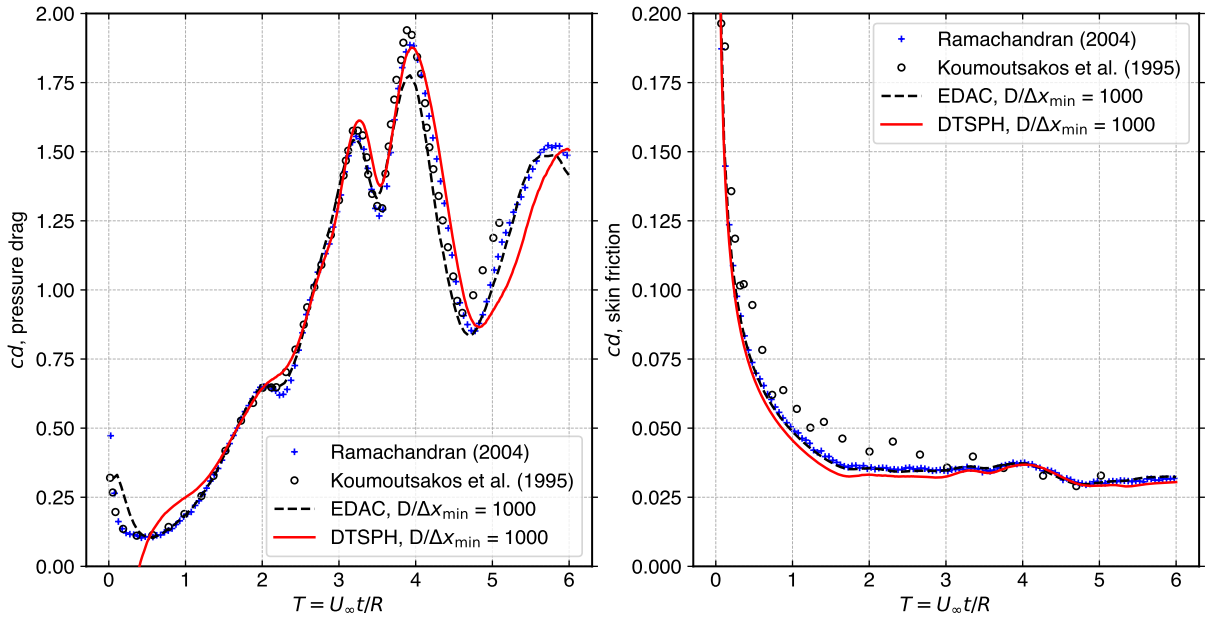


Figure 6.2: Comparison of Adaptive-DTSPH to with Adaptive-EDAC for flow past cylinder at $Re = 9500$. On the left is the coefficient of pressure drag and on the right is the coefficient of skin-friction drag.

6.3.2 Flow Past C-Shape

The flow past C-shape is simulated with the specifications given in Section 5.5.5. We use a tolerance of 10^{-3} for convergence in the iterations of DTSPH scheme. We use a minimum resolution of 20 and a maximum resolution of 200 with a Cr of 1.08, this gives 175k fluid particles in the domain. We do not study solution-adaptivity here as was done in the previous chapter. Figure 6.4 shows the coefficients of drag and lift for the flow past C-shape problem. The results are compared with the established results of the literature, also shown are the results obtained by the Adaptive-EDAC scheme used in the previous sections. The drag given by dual-time SPH has higher peaks than the results of EDAC but show good agreement in the overall features.

6.3.3 Performance Analysis

All the cases that are simulated in this chapter and the previous chapter are run on a AMD Ryzen 9 3950X 16-Core Processor. The PySPH framework (Ramachandran et al. 2021b) with OpenMP

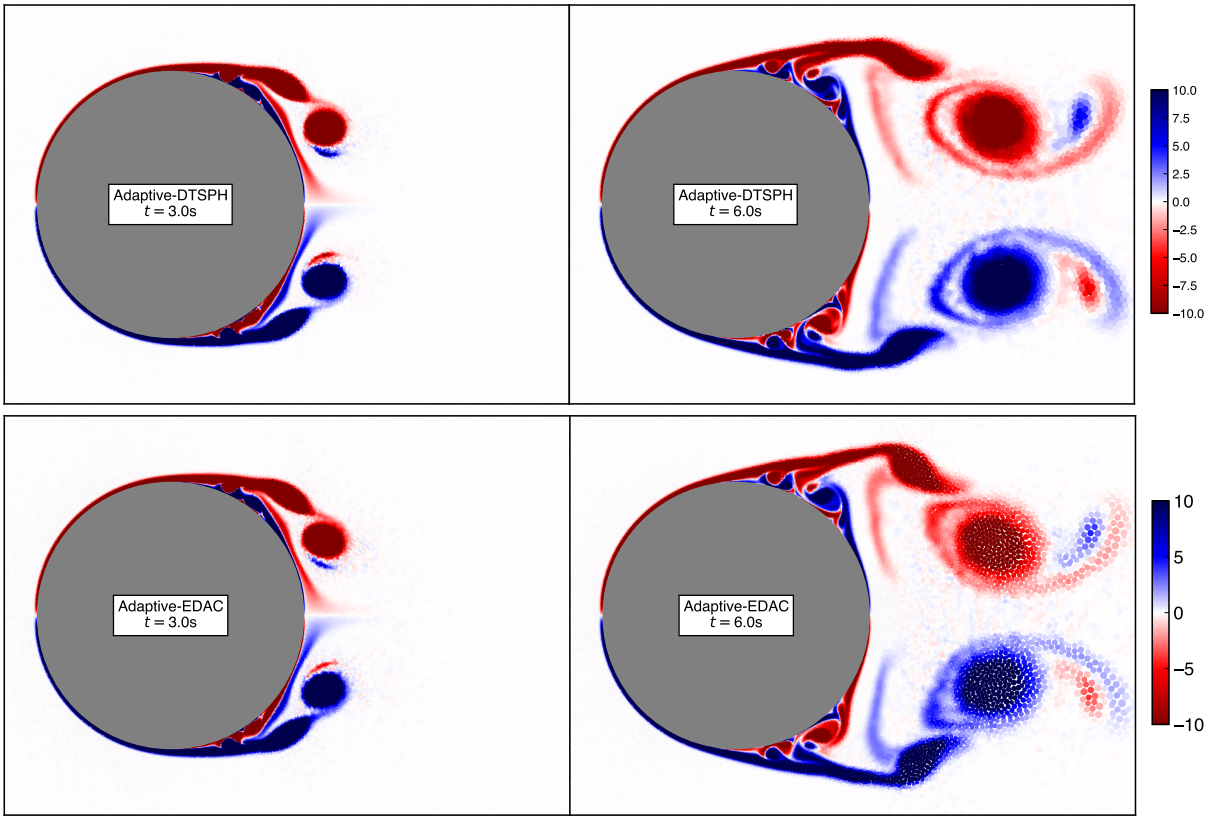


Figure 6.3: Vorticity distribution for flow past circular cylinder at *Re* 9500 at times of 3.0 s and 6.0 s. The maximum resolution used is 1000 with a Cr of 1.08 for Adaptive-DTSPH (top) and Cr of 1.15 for Adaptive-EDAC (bottom).

interface utilizes all the 16-Cores with hyper-threading enabled. The times shown in this chapter are the wall clock time from the start to end of the simulations.

Table 6.1 shows the performance of the Adaptive-DTSPH SPH scheme along with Adaptive-EDAC SPH scheme. Similar domain setup is used for both the schemes. For the flow past circular cylinder, we compare the case where the minimum resolution is 4 and the maximum resolution is 500, with a Cr of 1.08 giving a total fluid particles of approximately 216,000 in the Adaptive-EDAC and 230,000 in the Adaptive-DTSPH. A tolerance of 10^{-3} is used for the iteration in the DTSPH scheme. The performance of Adaptive-DTSPH is 3.24x faster than the EDAC scheme. For the flow past C-shape problem, we use a minimum resolution of 200 and a maximum resolution of 20 with Cr of 1.08 giving a total fluid particles of approximately 175,000 in both Adaptive-EDAC and Adaptive-DTSPH. The Adaptive-DTSPH scheme with a

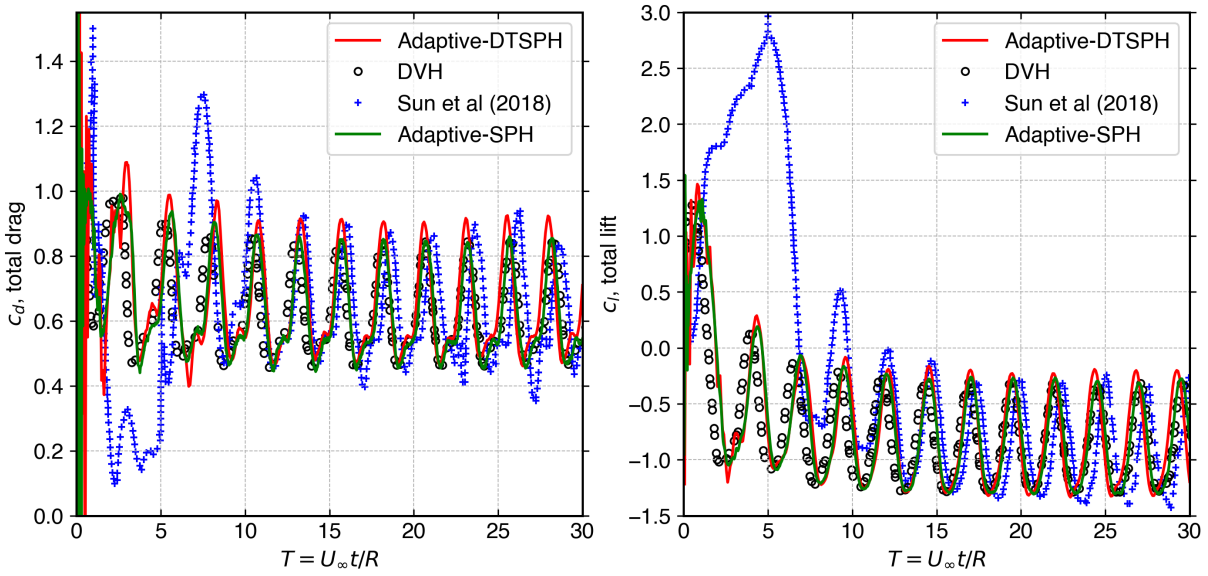


Figure 6.4: Comparison of Adaptive dual-time SPH with Adaptive EDAC-SPH. The coefficient of drag (left) and lift (right) for the flow past C-shape simulation at $Re = 2000$ is shown. Also shown are the results of Rossi et al. (2015) and Sun et al. (2018).

tolerance of 10^{-3} performs 9.02x faster than EDAC, this is a significant improvement. We note the timing of Adaptive-EDAC includes the background particle setup and its update every 200 time-steps, however, this only contributes roughly 5 to 7.5% of overall time.

Table 6.1: Comparison of CPU times between adaptive EDAC SPH and adaptive dual-time SPH schemes.

Scheme	Flow past cylinder ($D/\Delta x_{\min} = 500$)	Flow past C-shape ($D/\Delta x_{\min} = 200$)
Adaptive-EDAC SPH	5.025 hrs.	17.43 hrs.
Adaptive-DTSPH, $\epsilon = 10^{-3}$	1.55 hrs.	1.92 hrs.

6.4 Comparison with OpenFOAM

We now draw a comparison between the adaptive DTSPH method developed in this work with the open-source finite volume solver, OpenFOAM (Weller et al. 1998). We consider a simulation

of a heaving and pitching NACA0012 airfoil at a Reynolds number of 1000, as described in Bose et al. (2021). For the OpenFOAM simulation, we use the overset mesh to track the motion of the airfoil. Overset mesh (OpenFOAM 2023) is a robust addition to OpenFOAM to simulate cases with large relative motion. The domain setup for the problem is given in fig. 6.5. We use a chord length of 1 m, denoted by D, for the airfoil. The airfoil heaves in the y direction while pitching about the center of gravity, which is at a distance of 1/3rd chord from the nose of the airfoil. The equations for the motion are given by,

$$y(t) = A_0 \sin(2\pi f t), \quad \alpha(t) = \alpha_0 \sin(2\pi f t) \tag{6.3}$$

where, $y(t)$ is motion of the center of gravity, $\alpha(t)$ is the angle made by the chord about the x-axis in the clockwise direction, $A_0 = 0.6m$, the frequency of the motion f is $1/\pi$ s^{-1}, the time period $T = 1/f$, and $\alpha_0 = 15°$. We run the adaptive DTSPH simulation at four different minimum resolutions, $D/\Delta x_{\text{min}}$, of 50, 100, 200, and 400. The maximum resolution, $D/\Delta x_{\text{max}}$ is held fixed at 2. We use a Cr of 1.15 and the free-stream velocity from the inlet is moving in the x-direction by 1 ms^{-1}. The OpenFOAM simulation utilizes an overset mesh with 4 levels of refinement.

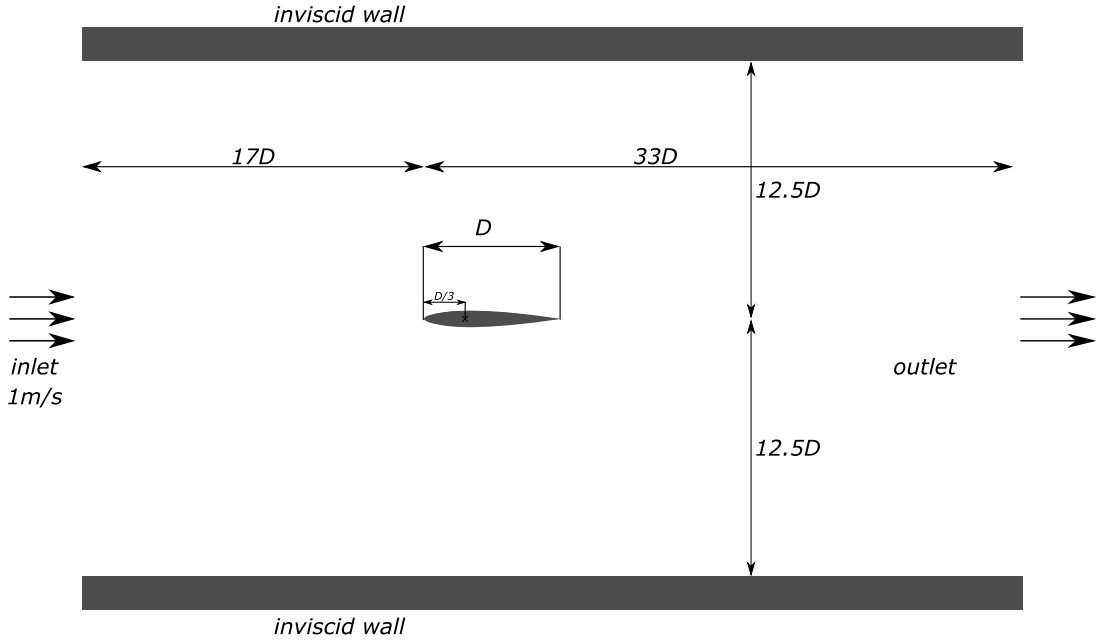


Figure 6.5: The domain setup for the pitching and heaving airfoil at *Re* 1000.

These refinement levels range from $D/\Delta x_{\text{max}} = 6.25$ to $D/\Delta x_{\text{min}} = 100$, where the resolution doubles from one level to another. To ensure accurate capturing of vortices, the trailing edge

is further refined with two additional levels, ranging from $D/\Delta x_{\min} = 100$ to $D/\Delta x_{\min} = 400$. For the simulation, the solver `overPimpleDyMFoam` is employed, and adaptive time-stepping is used.

Figure 6.6 shows the time evolution of the coefficients of lift and drag with various minimum resolutions. The resolutions of 100, 200, and 400 show a good match for the lift coefficient, although there is some variation in the drag coefficient. Additionally, the resolution of 50 exhibits the characters but shows higher variability in both coefficients.

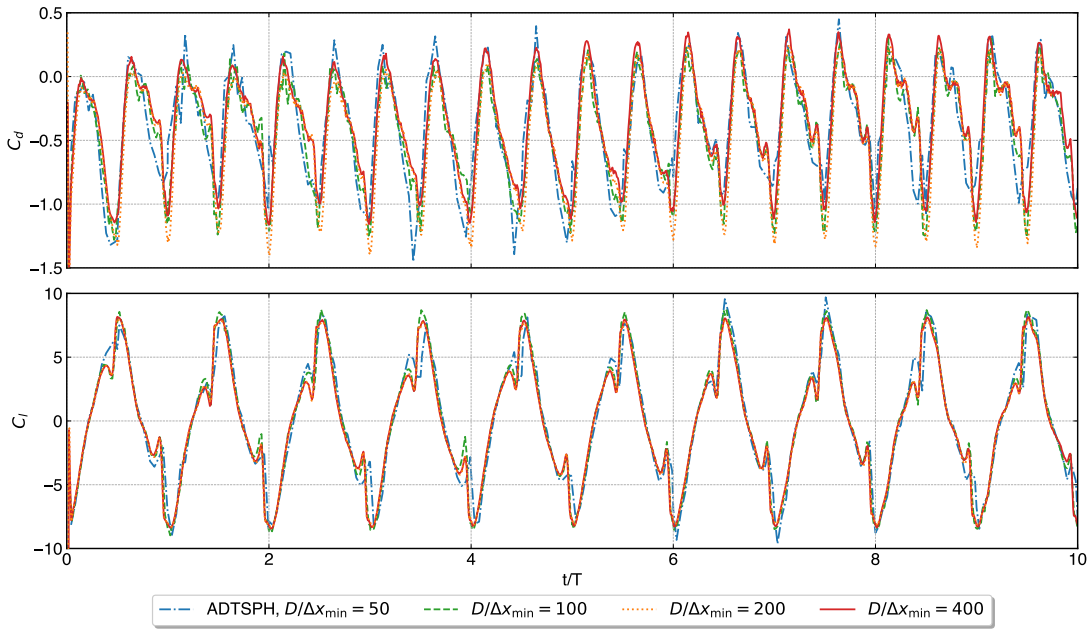


Figure 6.6: The time evolution of lift and drag coefficients for a pitching, and heaving airfoil with varying resolution simulated by adaptive DTSPH.

Figure 6.7 shows the comparison of adaptive DTSPH with a resolution of $D/\Delta x_{\min} = 200$ with the OpenFOAM simulation. The lift coefficient agrees well, but the drag coefficient shows a phase shift. The OpenFOAM results have a high-frequency noise imposed on the trend, which may be attributed to the interpolation of properties between meshes of different sizes. The adaptive DTSPH results do not vary much between successive periods, whereas the OpenFOAM results display noticeable differences. Figure 6.8 show the vorticity distribution at three different time instances. Table 6.2 summarizes the performance of adaptive DTSPH and OpenFOAM with overset mesh executed on the AMD Ryzen 9 3950X 16-Core Processor. As the resolution increases, the time taken approximately doubles, going from the resolution of 50 to 100 and 100

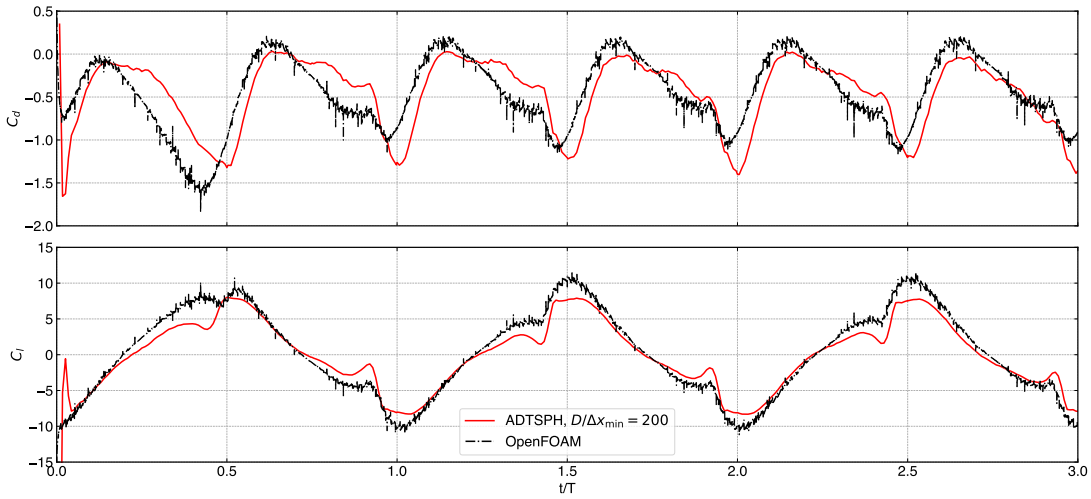


Figure 6.7: Comparison of lift and drag coefficients of adaptive DTSPH ($D/\Delta x_{\text{min}} = 200$) with OpenFOAM (with overset mesh).

to 200. This can be attributed to the halving of the time-step with an increase in the resolution, while the total number of particles does not vary linearly with the resolution. Moreover, in simulations with a resolution of 400, the time taken is four times longer than with a resolution of 200. This is primarily due to the nearly doubled number of particles and halving of the time-step. The OpenFOAM simulation employs adaptive time stepping, where the average resolution, $D/\Delta x_{\text{min}}$, is approximately 100 near the leading edge and increases up to 400 at the trailing edge. This configuration results in a total of 43,000 cells. The OpenFOAM simulation completes in nearly 25% less time compared to the SPH simulation with $D/\Delta x_{\text{min}} = 200$. The adaptive DTSPH solution with $D/\Delta x_{\text{min}} = 100$ also exhibit good results which are less noisy and is about $2\times$ faster than the OpenFOAM simulation. Considering the significant amount of time required for the mesh setup, we believe for complex motion scenarios, the adaptive SPH shows promising performance benefits over grid-based solvers. Furthermore, the solution is devoid of noise.

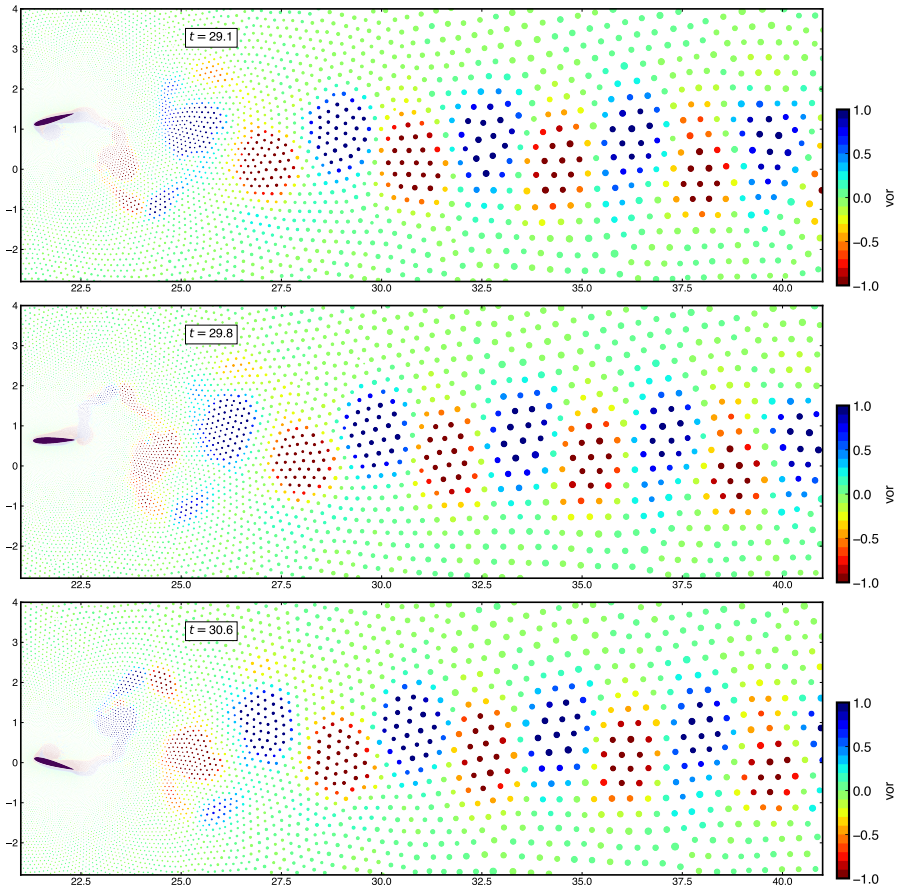


Figure 6.8: Vorticity distribution, with the point size corresponding to the mass of the particle, simulated by adaptive DTSPH ($D/\Delta x_{\min} = 400$). Three different time instances corresponding to 9.25π (top), 9.5π (mid), and 9.75π (bottom) are shown.

Scheme	Time (h)	Total no. of particles
ADTSPH, $D/\Delta x_{\min} = 50$	1.35	28k
ADTSPH, $D/\Delta x_{\min} = 100$	3.14	32k
ADTSPH, $D/\Delta x_{\min} = 200$	8.23	47k
ADTSPH, $D/\Delta x_{\min} = 400$	23.79	80k
OpenFOAM, with overset mesh	6.35	43k cells

Table 6.2: Performance of adaptive DTSPH and OpenFOAM for the pitching and heaving airfoil.

6.5 Summary

We have proposed a promising fast, adaptive dual-time stepping scheme for weakly-compressible flow simulations. We have incorporated the variable-h correction terms (Vacondio et al. 2012a) to the DTSPH scheme. The adaptive algorithm, without employing background particles, remains the same, as elaborated in the previous chapter. The accuracy is demonstrated by solving the flow past a circular cylinder and C-shape problems. We also verified that the Adaptive-DTSPH outperforms the traditional weakly-compressible EDAC SPH method by 3 to 9x times. We have shown that for a complex motion low-speed aerodynamics problem adaptive DTSPH can potentially compete with OpenFOAM.

7 Concluding Remarks

> *It's difficult to describe to a lay person that wonderful, almost magical moment of revelation in the solution of a problem or in the understanding of a concept. The problem or concept need not be grandiose, or even important, and often it is forgotten the next day. But that seems unimportant.*
>
> — Morton Gurtin, *Timoshenko medal lecture*

We have developed the fast methods, solved a range of test cases to provide the numerical evidence, and have shown the computational efficiency of the methods. We introduced a simple, iterative, incompressible SPH scheme that is based on the original projection formulation of Cummins et al. (1999). The method is matrix-free, fast, and suitable for execution on GPUs. The formulation is simple and easy to implement. We proposed a novel technique to ensure a homogeneous distribution of particles. In addition, we developed a modified solid wall boundary condition and show how we can implement simple inlet and outlet boundary conditions. We demonstrate the accuracy and efficiency of the new scheme with a suite of benchmark problems in two and three dimensions involving internal and external flows.

We carefully derive the dual-time SPH scheme and show that it is robust and accurate through several benchmarks in two and three dimensions. We also show that the scheme produces results that are as accurate as the δ-SPH scheme (Antuono et al. 2010) as well as the EDAC scheme (Ramachandran et al. 2019) while being up to seven times faster. The method is matrix-free and may be implemented in the context of any explicit SPH scheme. The DTSPH method does introduce a few new parameters in the form of the term $\beta = \Delta t/\Delta\tau$ and the tolerance ϵ. We discuss how these parameters can be set based on rational considerations. For a reasonable

choice of parameters, the performance of the method is comparable to that of incompressible SPH schemes.

We have established an adaptive particle method with a weakly-compressible scheme. Recent trends in high performance computing is towards parallel processing. Parallel processing is ubiquitous and developing parallel algorithms is a deciding factor in choosing a particular technique. Our adaptive particle method is fully parallel and the fast algorithms, too, are fully parallel. Note that they are only parallel in a shared memory sense, and we have not shown this for distributed memory, however, that should be possible. This makes the SPH method a serious contender in challenging fluid flow applications.

We perform simulations at unprecedented resolution in SPH for the flow past a circular cylinder for a variety of Reynolds numbers in the range 40 to 9500. For example, at $Re = 9500$ we use a resolution of $D/\Delta x_{\text{min}} = 1000$ and $D/\Delta x_{\text{max}} = 4$ giving a ratio of length scales of 250. This requires 16 levels of refinement with a domain size of $25D \times 15D$, requiring only 200,000 particles. The results are in good comparison with that of (Koumoutsakos et al. 1995; Ramachandran 2004) who also employ similar number of particles. This shows the effectiveness and accuracy of the adaptive resolution method.

For a Reynolds number of 3000 with a $D/\Delta x_{\text{min}} = 50$ we are able to obtain similar accuracy with the adaptive refinement using 30 times less particles, with a 25-fold speed improvement when compared with that using a fixed resolution.

We also show how the method can be used to simulate CFD problems that involve stationary and moving geometries of varying complexity. The source code is open-source and can be obtained from `https://gitlab.com/pypr/asph_motion`. To our knowledge this is the only open-source code available for adaptive SPH. The proposed adaptive algorithm is executed before any SPH computations are performed at every time-step. This makes it easy to incorporate into an existing non-adaptive EDAC-SPH solver.

The integration of adaptive particle refinement strategy with the formulation of dual-time SPH formulation (Ramachandran et al. 2021a) shows encouraging results. The test cases show up to 3 to 9x speed-up in computation time.

7.1 Current Limitations & Future Vision

Despite achieving satisfactory results, our scheme does not exhibit second-order convergence. Recently, a new family of second-order convergent schemes (Negi et al. 2021b) has been identified by carefully choosing the form of SPH operators used in the discretization process. We plan to incorporate these changes into our dual-time SPH formulation and establish second-order accuracy.

There is currently no existing study in the literature that analyzes the nature of the error introduced in the flow due to the presence of variable-resolution particles. The method of manufactured solutions is an exciting new alternative in the SPH community. In Negi et al. (2021e), the authors analyzed the convergence and accuracy of weakly-compressible SPH by manufacturing solutions. This is an alternative to having an analytical solution and is even more advantageous due to the control it provides. We plan to study the convergence and accuracy of adaptive particle refinement using the method of manufacturing solutions.

The promising outcomes demonstrated by the Adaptive-DTSPH approach suggest a potential avenue for future research in applications to solid mechanics problems. A method of this kind would provide significant computational advantages. This would also be of considerable importance in the area of fluid-structure interaction, which is an important emerging area of investigation with the SPH method.

We would like to extend the adaptive strategy to time discretization. A single global time step governs our simulations, and the finest particle resolution dictates this time step. This poses a severe impediment to improving the performance of the computation. The adaptive time-slicing strategy is based on a local time step, where a group of particles is given a time based on the finest resolution in the group. There will be several batches of particles, and the interaction between these will be synchronized to reduce the error. This will impact the computational time significantly. This is a new avenue in SPH. There have been several works in the mesh-based methods (Neal et al. 1989; Springel 2010a; Wu et al. 2000), a few from the computer graphics community (Ihmsen et al. 2010; Reinhardt et al. 2017) are using adaptive-time stepping for SPH but a systematic study is still lacking in SPH.

Comparison of SPH with the finite volume open-source code of OpenFOAM is a valuable study, especially in the cases where the mesh distortion poses severe limitations. As demonstrated in the previous chapter, handling complex motions of complex geometries is relatively straightforward in SPH, unlike in mesh-based methods. However, SPH lacks in a few areas, like

turbulence modeling, high Reynolds number simulations, and CFL time-step restrictions, where the mesh-based alternatives could excel. A comparative study between SPH and OpenFOAM studying several problems showcasing the differences could be helpful and is currently ongoing. Preliminary simulations suggest that SPH is as fast as OpenFOAM, and when factoring in mesh generation and setup time, SPH could be a better choice. As we encounter more complex fluid flow problems, such as those involving free surfaces, sloshing, and multiphase flows, SPH may potentially outperform traditional mesh-based methods while being far easier to set up.

Lastly, we plan to extend our adaptive Dual-time SPH method to three dimensions. Although developing the adaptive split and merge algorithm to three dimensions is straightforward, we anticipate challenges in addressing inaccuracies in the viscous operator, particularly in moderate to high Reynolds number flows, boundary handling, and turbulence modeling. The method of manufactured solutions shows a promising path to tackle the inaccuracies in the viscous operator and the boundary handling.

Our goal is to promote the use of SPH in computational fluid dynamics areas where mesh-based schemes are inconvenient or unsuitable. To this end, we have developed a fast, efficient, accurate, and adaptive SPH method that can be applied to a wide range of fluid flow problems.

Our research, combined with recent developments in the field, has demonstrated the increasing popularity of SPH in the landscape of fluid dynamics. SPH has proven particularly useful in situations where traditional mesh-based methods encounter difficulties, such as large-scale flood simulations, sloshing, and low-speed aerodynamics with complex motion. With our current SPH setup, we expect to rival OpenFOAM in low-speed aerodynamics scenarios involving complex motion.

Appendix I: SISPH Implementation Details

Here we outline the procedure in solving the iterative formulation using the PySPH framework (Ramachandran et al. 2021b). PySPH is a Python framework which implements various SPH formulations. The user code is written in Python from which high performance serial (OpenMP) or parallel (OpenCL or CUDA C) code is automatically generated. PySPH also supports multi-CPU execution using MPI.

Listing 1 shows the Python code used to define the inter-particle interactions for a Taylor-Green vortex. This is done in two separate stages. Each stage is defined as a list of equations. Each equation has a `dest` and `sources` keyword argument. The `dest` refers to the particles on which the equation is to be solved and `sources` is the list particles that influence the `dest` particles. `stage1` is a list of equations that are to be solved before the first integration step similarly `stage2` are solved before the second integration step. A `Group` is a PySPH construct which updates all the particles with a given set of equations. A `Group` is solved using the current particle properties. For example, in the `stage1`, the summation density is evaluated and updated density is found for all the particles before moving on to the next group of equations. In the listing 1, we see that the first stage performs the steps 2 to 4 in algorithm 1. The second stage performs the remaining computations. Step 5 in the algorithm is performed by the `VelocityDivergence` equation, next the equations of steps 7 and 8 are performed in an iterated group. The iterated group iterates until the convergence criterion is satisfied or if predefined maximum iterations are completed (in practice this limit is never reached). Finally, the last group performs steps 9

and 10. A suitable integrator updates the velocities and positions. More details on PySPH are available in (Ramachandran et al. 2021b).

An example equation is shown in listing 2. This shows how the equation for the pressure coefficients, i.e. eqs. (3.25) and (3.26), are written. The `initialize` method of the class initializes the variables to zero, `d_idx` is the index of `dest` particles. The `loop` method is called for every pairwise interaction with the neighbors, where `s_idx` is the index of the source (neighbor) particles. The source particle properties are prefixed with `s_` and the destination with `d_`, for example `s_pk` is an array of the pressure of the source particles at the k^{th} iteration, p_k, and `d_diag` is the diagonal term, D_{ii} of the destination particle. Various quantities like `DWIJ` which is the gradient vector for the current particle, and `XIJ` is the distance between destination particle and source particle are available to each method. The listings demonstrate the ease with which the new scheme can be implemented in the PySPH framework.

Listing 1: Algorithm in PySPH for a Taylor-Green simulation, showing two stages which are executed before the integration steps are performed.

```
stage1 = [
    Group(equations=[
        SummationDensity(dest='fluid', sources=['fluid'])
    ]),
    Group(equations=[
        LaminarViscosity(dest='fluid', sources=['fluid'],
                         nu=0.01,
        ),
        MomentumEquationArtificialStress(
            dest='fluid', sources=['fluid'], dim=2
        )
    ])]

stage2 = [
    Group(equations=[
        VelocityDivergence(dest='fluid', sources=['fluid'])
    ]),
    Group(equations=[
        Group(equations=[
            PressureCoeffs(dest='fluid', sources=['fluid']),
            PPESolve(dest='fluid', sources=['fluid'],
                     omega=0.5, tolerance=0.01)
```

```
        ],
        iterate=True, max_iterations=1000, min_iterations=2)
    ]),
    Group(equations=[
        MomentumEquationPressureGradient(
            dest='fluid', sources=['fluid']
        ),
        GTVFAcceleration(dest='fluid', sources=['fluid'],
                         pref=100.0)
    ])]
```

Listing 2: An example of equation in PySPH framework showing the implementation of diagonal and off-diagonal terms in the pressure solver.

```
class PressureCoeffs(Equation):
    def initialize(self, d_idx, d_diag, d_odiag):
        d_diag[d_idx] = 0.0
        d_odiag[d_idx] = 0.0

    def loop(self, d_idx, s_idx, s_m, d_rho, s_rho, d_diag,
             d_odiag, s_pk, XIJ, DWIJ, R2IJ, EPS):
        rhoij = (s_rho[s_idx] + d_rho[d_idx])
        rhoij2_1 = 1.0/(d_rho[d_idx]*rhoij)

        xdotdwij = (XIJ[0]*DWIJ[0] + XIJ[1]*DWIJ[1]
                    + XIJ[2]*DWIJ[2])

        fac = 4.0*s_m[s_idx]*rhoij2_1 * xdotdwij / (R2IJ + EPS)

        d_diag[d_idx] += fac
        d_odiag[d_idx] += -fac * s_pk[s_idx]
```

Appendix II: Derivation of DTSPH Perturbation Velocity

This section provides a derivation of the perturbation velocity that is given in eq. (4.17).

Using trapezoidal rule for integration, the displacement of the particle in pseudo time from the initial state $(\mathbf{x}^{k=0}, \mathbf{u}^{k=0}, 0)$ to the current pseudo time state $(\mathbf{x}^{k}, \mathbf{u}^{k+1}, \Delta t)$ when $k \rightarrow \infty$ is given by,

$$\mathbf{x}^{k+1} - \mathbf{x}^{n} = \Delta t \frac{(\mathbf{u}^{k+1} + \mathbf{u}^{n})}{2}, \tag{1}$$

similarly, using the trapezoidal rule of integration for the displacement between states in pseudo time is given by,

$$\mathbf{x}^{k+1} = \mathbf{x}^{k} + \frac{\Delta \tau}{2}(\hat{\mathbf{u}}^{k} + \hat{\mathbf{u}}^{k+1}). \tag{2}$$

Expanding $\mathbf{x}^{k}$ in terms of $\mathbf{x}^{0}$ (i.e. $k = 0$) is,

$$\mathbf{x}^{k+1} = \mathbf{x}^{0} + \frac{\Delta \tau}{2}\hat{\mathbf{u}}^{0} + \Delta \tau \sum_{j=1}^{k} \hat{\mathbf{u}}^{j} + \frac{\Delta \tau}{2}\hat{\mathbf{u}}^{k+1}, \tag{3}$$

where, the position before pseudo time iteration $\mathbf{x}^{0}$ is given by eq. (4.11).

Substitute the above eq. (3) into the eq. (1) we get,

$$\mathbf{x}^{0} - \mathbf{x}^{n} + \frac{\Delta \tau}{2}(\hat{\mathbf{u}}^{0} + \hat{\mathbf{u}}^{k+1}) + \Delta \tau \sum_{j=1}^{k} \hat{\mathbf{u}}^{j} = \Delta t \frac{(\mathbf{u}^{k+1} + \mathbf{u}^{n})}{2}. \tag{4}$$

Rearranging terms to get $\mathbf{u}^{k+1}$ as,

$$\mathbf{u}^{k+1} = \frac{2}{\Delta t}\left(\mathbf{x}^0 - \mathbf{x}^n\right) + \frac{\Delta\tau}{\Delta t}(\hat{\mathbf{u}}^0 + \hat{\mathbf{u}}^{k+1}) + \frac{2\Delta\tau}{\Delta t}\sum_{j=1}^{k}\hat{\mathbf{u}}^j - \mathbf{u}^n. \tag{5}$$

Similarly $\mathbf{u}^k$ is written as,

$$\mathbf{u}^{k} = \frac{2}{\Delta t}\left(\mathbf{x}^0 - \mathbf{x}^n\right) + \frac{\Delta\tau}{\Delta t}(\hat{\mathbf{u}}^0 + \hat{\mathbf{u}}^{k}) + \frac{2\Delta\tau}{\Delta t}\sum_{j=1}^{k-1}\hat{\mathbf{u}}^j - \mathbf{u}^n. \tag{6}$$

Subtract eq. (6) from eq. (5),

$$\mathbf{u}^{k+1} - \mathbf{u}^{k} = \frac{\Delta\tau}{\Delta t}(\hat{\mathbf{u}}^{k+1} + \hat{\mathbf{u}}^{k}). \tag{7}$$

By substituting (eq. (4.14)) we get,

$$\hat{\mathbf{u}}^k = \frac{\Delta t}{\Delta\tau}(\mathbf{u}^{k+1} - \mathbf{u}^k) = \Delta t\left(\frac{d\mathbf{u}}{d\tau}\right)^{k+1/2}. \tag{8}$$

Note that in the limit $k \to \infty$, $\hat{\mathbf{u}}^{k+1}$ goes to zero, and $\lim_{k\to\infty}\mathbf{u}^{k+1} = \mathbf{u}^{n+1}$.

Bibliography

[1] S. Adami, X.Y. Hu, and N.A. Adams. "A generalized wall boundary condition for smoothed particle hydrodynamics". In: *Journal of Computational Physics* 231.21 (Aug. 2012), pp. 7057–7075 (cit. on pp. 20, 23, 31, 32, 64, 100).

[2] S. Adami, X.Y. Hu, and N.A. Adams. "A transport-velocity formulation for smoothed particle hydrodynamics". In: *Journal of Computational Physics* 241 (May 2013), pp. 292–307 (cit. on pp. 4, 7, 19, 24, 34, 41, 52, 56, 68, 76, 97).

[3] Bart Adams et al. "Adaptively Sampled Particle Fluids". In: *ACM Transactions on Graphics* 26.3 (2007), p. 8 (cit. on pp. 6, 92).

[4] Dinesh Adepu and Prabhu Ramachandran. "A corrected transport-velocity formulation for fluid and structural mechanics with SPH". In: *arXiv e-prints*, arXiv:2106.00756 (May 2021), arXiv:2106.00756. arXiv: 2106.00756 [physics.flu-dyn] (cit. on pp. 7, 20, 97, 151).

[5] M. Antuono et al. "Free-surface flows solved by means of SPH schemes with numerical diffusive terms". In: *Computer Physics Communications* 181.3 (2010), pp. 532–549. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2009.11.002 (cit. on pp. 4, 7, 56, 62, 68, 91, 165).

[6] Lorena A. Barba. "The hard road to reproducibility". In: *Science* 354.6308 (Oct. 2016), pp. 142–142. DOI: 10.1126/science.354.6308.142 (cit. on p. 11).

[7] D A Barcarolo. “Improvement of the precision and the efficiency of the SPH method: theoretical and numerical study”. Theses. Ecole Centrale de Nantes (ECN), Oct. 2013 (cit. on pp. 3, 7, 22, 27, 29, 42).

[8] D. A. Barcarolo et al. “Adaptive particle refinement and derefinement applied to the smoothed particle hydrodynamics method”. In: *Journal of Computational Physics* 273 (Sept. 2014), pp. 640–657. ISSN: 0021-9991 (cit. on pp. 5, 8).

[9] DA Barcarolo, D Le Touzé, and F de Vuyst. “Validation of a new fully-explicit incompressible Smoothed Particle Hydrodynamics method”. In: *Blucher Mechanical Engineering Proceedings* 1.1 (2014) (cit. on pp. 3, 6, 40).

[10] Mihai Basa, Nathan J. Quinlan, and Martin Lastiwka. “Robustness and accuracy of SPH formulations for viscous flow”. In: *International Journal for Numerical Methods in Fluids* 60 (2009), pp. 1127–1148 (cit. on p. 98).

[11] Hossein Basser, Murray Rudman, and Edoardo Daly. “SPH modelling of multi-fluid lock-exchange over and within porous media”. In: *Advances in Water Resources* 108 (2017), pp. 15–28. ISSN: 0309-1708 (cit. on pp. 3, 4, 22, 27, 29).

[12] Carl Boettiger. “An Introduction to Docker for Reproducible Research”. In: *SIGOPS Oper. Syst. Rev.* 49.1 (Jan. 2015), pp. 71–79. ISSN: 0163-5980 (cit. on p. 11).

[13] J. Bonet and T.-S.L. Lok. “Variational and momentum preservation aspects of Smooth Particle Hydrodynamic formulations”. In: *Computer Methods in Applied Mechanics and Engineering* 180.1 (1999), pp. 97–115. ISSN: 0045-7825 (cit. on p. 15).

[14] Chandan Bose, Sayan Gupta, and Sunetra Sarkar. “Dynamic interlinking between near- and far-field wakes behind a pitching–heaving airfoil”. In: *Journal of Fluid Mechanics* 911 (Jan. 2021). DOI: 10.1017/jfm.2020.1030 (cit. on p. 160).

[15] L. Chiron et al. “Analysis and improvements of Adaptive Particle Refinement (APR) through CPU time, accuracy and robustness considerations”. In: *Journal of Computational Physics* 354 (Feb. 2018), pp. 552–575. ISSN: 0021-9991 (cit. on pp. 5, 8, 92, 120, 121, 124, 126).

[16] A. J. Chorin and J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics*. Springer US, 1990 (cit. on p. 2).

[17] Alexandre Joel Chorin. “A numerical method for solving incompressible viscous flow problems”. In: *Journal of Computational Physics* 2.1 (1967), pp. 12–26. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(67)90037-X (cit. on pp. 10, 55).

[18] Alexandre Joel Chorin. “Numerical Solution of the Navier-Stokes Equations”. In: *Mathematics of Computation* 22.104 (1968), pp. 745–762. ISSN: 00255718, 10886842 (cit. on p. 29).

[19] Alex D. Chow et al. “Incompressible SPH (ISPH) with fast Poisson solver on a GPU”. In: *Computer Physics Communications* 226 (2018), pp. 81–103. ISSN: 0010-4655 (cit. on pp. 3, 22, 48).

[20] Jonathan R Clausen. “Entropically damped form of artificial compressibility for explicit simulation of incompressible flow”. In: *Physical Review E* 87.1 (Jan. 2013), pp. 013309-1–013309-12 (cit. on p. 16).

[21] Paul W Cleary and Joseph J Monaghan. “Conduction Modelling Using Smoothed Particle Hydrodynamics”. In: *Journal of Computational Physics* 148.1 (1999), pp. 227–264. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1998.6118 (cit. on pp. 17, 99).

[22] Andrea Colagrossi. “A meshless Lagrangian method for free-surface and interface flows with fragmentation”. In: *These, Universita di Roma* (2005) (cit. on pp. 43, 79).

[23] G Colicchio, M Greco, and OM Faltinsen. “Fluid-body interaction on a Cartesian grid: dedicated studies for a CFD validation”. In: *Proc. 21st International Workshop on Water Waves and Floating Bodies, IWWWFB* 6 (2006) (cit. on pp. 144–148).

[24] Alejandro Jacobo Cabrera Crespo. “Application of the smoothed particle hydrodynamics model SPHysics to free-surface hydrodynamics”. ENGLISH. PhD thesis. Universidade de Vigo, 2008 (cit. on p. 19).

[25] S. J Cummins and M. Rudman. “An SPH projection method”. In: *Journal of Computational Physics* 152 (1999), pp. 584–607 (cit. on pp. 3, 4, 22–26, 35, 54, 165).

[26] Mathieu Desbrun and Marie-Paule Cani. *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. rep. 3829. INRIA, 1999, p. 26 (cit. on pp. 6, 92).

[27] D. Durante et al. “Numerical Simulations of the Transition from Laminar to Chaotic Behaviour of the Planar Vortex Flow Past a Circular Cylinder”. In: *Communications in Nonlinear Science and Numerical Simulation* 48 (July 2017), pp. 18–38. ISSN: 10075704. DOI: 10.1016/j.cnsns.2016.12.013 (cit. on pp. 137, 138).

[28] R. Fatehi and M.T. Manzari. “Error Estimation in Smoothed Particle Hydrodynamics and a New Scheme for Second Derivatives”. In: *Computers & Mathematics with Applications* 61.2 (Jan. 2011), pp. 482–498. ISSN: 08981221. DOI: 10.1016/j.camwa.2010.11.028 (cit. on p. 124).

[29] R. Fatehi et al. “Density-Based Smoothed Particle Hydrodynamics Methods for Incompressible Flows”. In: *Computers & Fluids* 185 (May 2019), pp. 22–33. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2019.02.018 (cit. on pp. 4, 56, 68, 91).

[30] J. Feldman and J. Bonet. “Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems”. In: *International Journal for Numerical Methods in Engineering* 72.3 (2007), pp. 295–324. ISSN: 1097-0207 (cit. on pp. 5, 9, 93, 103, 104).

[31] Angela Ferrari et al. “A New 3D Parallel SPH Scheme for Free Surface Flows”. In: *Computers & Fluids* 38.6 (June 2009), pp. 1203–1217. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2008.11.012 (cit. on p. 63).

[32] A. Ghasemi V., B. Firoozabadi, and M. Mahdinia. “2D Numerical Simulation of Density Currents Using the SPH Projection Method”. In: *European Journal of Mechanics - B/Fluids* 38 (Mar. 2013), pp. 38–46. ISSN: 09977546. DOI: 10.1016/j.euromechflu.2012.10.004 (cit. on p. 76).

[33] U. Ghia, K. N. Ghia, and C. T. Shin. “High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method”. In: *Journal of Computational Physics* 48 (1982), pp. 387–411 (cit. on pp. 41–44, 76, 78, 80, 81, 130, 131).

[34] R. A. Gingold and J. J. Monaghan. “Smoothed particle hydrodynamics: Theory and application to non-spherical stars”. In: *Monthly Notices of the Royal Astronomical Society* 181 (1977), pp. 375–389 (cit. on p. 1).

[35] Philip M. Gresho and Stevens T. Chan. “On the Theory of Semi-Implicit Projection Methods for Viscous Incompressible Flow and Its Implementation via a Finite Element Method That Also Introduces a Nearly Consistent Mass Matrix. Part 2: Implementation”. In: *International Journal for Numerical Methods in Fluids* 11.5 (Oct. 1990), pp. 621–659. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.1650110510 (cit. on p. 127).

[36] Asmelash Haftu, Abhinav Muta, and Prabhu Ramachandran. “Parallel Adaptive Weakly-Compressible SPH for Complex Moving Geometries”. In: *Computer Physics Communications* 277 (Aug. 2022), p. 108377. ISSN: 00104655. DOI: 10.1016/j.cpc.2022.108377 (cit. on pp. 12, 190).

[37] Lars Hernquist and Neal Katz. “TREESPH - A Unification of SPH with the Hierarchical Tree Method”. In: *The Astrophysical Journal Supplement Series* 70 (June 1989), p. 419. ISSN: 0067-0049, 1538-4365. DOI: 10.1086/191344 (cit. on pp. 94, 98).

[38] Gerhard A. Holzapfel. In: *Meccanica* 37.4/5 (2002), pp. 489–490. DOI: 10.1023/a:1020843529530 (cit. on p. 97).

[39] Philip F. Hopkins. “A New Class of Accurate, Mesh-Free Hydrodynamic Simulation Methods”. In: *Monthly Notices of the Royal Astronomical Society* 450.1 (June 2015), pp. 53–110. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stv195 (cit. on pp. 127, 128).

[40] Majid Hosseini, Mehrdad Manzari, and Siamak Hannani. “A fully explicit three-step SPH algorithm for simulation of non-Newtonian fluid flow”. In: *International Journal of Numerical Methods for Heat and Fluid Flow* 17 (Sept. 2007). DOI: 10.1108/09615530710777976 (cit. on pp. 3, 22, 27, 29, 40).

[41] Wei Hu et al. “A Consistent Spatially Adaptive Smoothed Particle Hydrodynamics Method for Fluid–Structure Interactions”. In: *Computer Methods in Applied Mechanics and Engineering* 347 (Apr. 2019), pp. 402–424. ISSN: 00457825. DOI: 10.1016/j.cma.2018.10.049 (cit. on pp. 5, 6, 93).

[42] X.Y. Hu and N.A. Adams. “An incompressible multi-phase SPH method”. In: *Journal of Computational Physics* 227.1 (Nov. 2007), pp. 264–278 (cit. on pp. 3, 24).

[43] J.P. Hughes and D.I. Graham. “Comparison of incompressible and weakly-compressible SPH models for free-surface water flows”. In: *Journal of Hydraulic Research* 48 (2010), pp. 105–117 (cit. on p. 64).

[44] Markus Ihmsen et al. “Boundary Handling and Adaptive Time-stepping for PCISPH”. In: *Workshop in Virtual Reality Interactions and Physical Simulation ”VRIPHYS” (2010)*. Ed. by Kenny Erleben, Jan Bender, and Matthias Teschner. The Eurographics Association, 2010. ISBN: 978-3-905673-78-4. DOI: 10.2312/PE/vriphys/vriphys10/079-088 (cit. on p. 167).

[45] Markus Ihmsen et al. “Implicit Incompressible SPH”. In: *IEEE Trans. Vis. Comput. Graph.* 20.3 (2014), pp. 426–435 (cit. on p. 4).

[46] Abbas Khayyer and Hitoshi Gotoh. “Enhancement of Performance and Stability of MPS Mesh-Free Particle Method for Multiphase Flows Characterized by High Density Ratios”. In: *Journal of Computational Physics* 242 (June 2013), pp. 211–233. ISSN: 00219991. DOI: 10.1016/j.jcp.2013.02.002 (cit. on pp. 79, 81).

[47] S. Koshizuka and Y. Oka. “Moving-Particle Semi-Implicit method for fragmentation of incompressible fluid”. In: *Nuclear Science and Engineering* 123 (1996), pp. 421–434 (cit. on pp. 46, 48, 84, 89).

[48] P. Koumoutsakos and A. Leonard. “High-Resolution Simulations of the Flow around an Impulsively Started Cylinder Using Vortex Methods”. In: *Journal of Fluid Mechanics* 296 (Aug. 1995), pp. 1–38. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/S0022112095002059 (cit. on pp. 120, 132, 133, 135, 147, 148, 166).

[49] Martin Lastiwka, Mihai Basa, and Nathan J. Quinlan. “Permeable and Non-Reflecting Boundary Conditions in SPH”. In: *International Journal for Numerical Methods in Fluids* 61.7 (Nov. 2009), pp. 709–724. ISSN: 02712091, 10970363. DOI: 10.1002/fld.1971 (cit. on pp. 21, 101, 132).

[50] E.-S. Lee et al. “Comparisons of Weakly Compressible and Truly Incompressible Algorithms for the SPH Mesh Free Particle Method”. In: *Journal of Computational Physics* 227.18 (Sept. 2008), pp. 8417–8436. ISSN: 00219991. DOI: 10.1016/j.jcp.2008.06.005 (cit. on pp. 76, 84).

[51] Eun-Sug Lee et al. “Application of weakly compressible and truly incompressible SPH to 3-D water collapse in waterworks”. In: *Journal of Hydraulic Research* 48.sup1 (2010), pp. 50–60. eprint: https://doi.org/10.1080/00221686.2010.9641245 (cit. on pp. 46, 47, 84).

[52] Benedict J. Leimkuhler, Sebastian Reich, and Robert D. Skeel. "Integration Methods for Molecular Dynamics". In: *Mathematical Approaches to Biomolecular Structure and Dynamics*. Ed. by Jill P. Mesirov, Klaus Schulten, and De Witt Sumners. New York, NY: Springer New York, 1996, pp. 161–185. ISBN: 978-1-4612-4066-2 (cit. on p. 18).

[53] Randall J. LeVeque. "Python Tools for Reproducible Research on Hyperbolic Problems". In: *Computing in Science & Engineering* 11.1 (2009), pp. 19–27. DOI: 10.1109/MCSE.2009.13 (cit. on p. 11).

[54] S.J. Lind, P.K. Stansby, and B.D. Rogers. "Incompressible–Compressible Flows with a Transient Discontinuous Interface Using Smoothed Particle Hydrodynamics (SPH)". In: *Journal of Computational Physics* 309 (Mar. 2016), pp. 129–147. ISSN: 00219991. DOI: 10.1016/j.jcp.2015.12.005 (cit. on pp. 81, 84).

[55] S.J. Lind et al. "Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves". In: *Journal of Computational Physics* 231.4 (2012), pp. 1499–1523. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2011.10.027 (cit. on pp. 7, 19, 24, 35, 53, 80, 99, 111, 117, 151).

[56] Richard Liska and Burton Wendroff. "Comparison of Several Difference Schemes on 1D and 2D Test Problems for the Euler Equations". In: *SIAM Journal on Scientific Computing* 25.3 (Jan. 2003), pp. 995–1017. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/S1064827502402120 (cit. on p. 127).

[57] Wen-Bin Liu et al. "Application of smoothed particle hydrodynamics method for simulating the flooding process of a damaged ship cabin in full-time domain". In: *Ocean Engineering* 248 (Mar. 2022), p. 110716 (cit. on p. 9).

[58] L. B. Lucy. "A numerical approach to testing the fission hypothesis". In: *The Astronomical Journal* 82.12 (1977), pp. 1013–1024 (cit. on p. 1).

[59] S. Marrone et al. "δ-SPH model for simulating violent impact flows". In: *Computer Methods in Applied Mechanics and Engineering* 200 (Mar. 2011), pp. 1526–1542. DOI: 10.1016/j.cma.2010.12.016 (cit. on pp. 3, 7, 84).

[60] S. Marrone et al. "An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers". In: *Journal of Computational Physics* 245 (2013), pp. 456–475. ISSN: 0021-9991 (cit. on p. 51).

[61] Salvatore Marrone et al. “An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers”. In: *Journal of Computational Physics* 245 (2013), pp. 456–475. DOI: 10.1016/j.jcp.2013.03.011 (cit. on pp. 144, 145).

[62] J. E. Marsden and M. West. “Discrete mechanics and variational integrators”. In: *Acta Numerica* 10 (2001), pp. 357–514. DOI: 10.1017/S096249290100006X (cit. on p. 18).

[63] Olivier Mesnard and Lorena A. Barba. “Reproducible and Replicable Computational Fluid Dynamics: It’s Harder Than You Think”. In: *Computing in Science & Engineering* 19.4 (2017), pp. 44–55. DOI: 10.1109/MCSE.2017.3151254 (cit. on p. 11).

[64] J. J. Monaghan. “Smoothed Particle Hydrodynamics”. In: *Annual Review of Astronomy and Astrophysics* 30.1 (1992), pp. 543–574. eprint: https://doi.org/10.1146/annurev.aa.30.090192.002551 (cit. on pp. 17, 25).

[65] J. J. Monaghan. “Simulating Free Surface Flows with SPH”. In: *Journal of Computational Physics* 110 (1994), pp. 399–406 (cit. on pp. 1, 4, 68, 81).

[66] J. J. Monaghan. “Smoothed Particle Hydrodynamics”. In: *Reports on Progress in Physics* 68 (2005), pp. 1703–1759 (cit. on pp. 16, 18, 25, 62, 98).

[67] Joseph P. Morris, Patrick J. Fox, and Yi Zhu. “Modeling low Reynolds number incompressible flows using SPH”. In: *Journal of Computational Physics* 136.1 (1997), pp. 214–226. DOI: http://dx.doi.org/10.1006/jcph.1997.5776 (cit. on p. 63).

[68] Abhinav Muta and Prabhu Ramachandran. “Efficient and accurate adaptive resolution for weakly-compressible SPH”. In: *Computer Methods in Applied Mechanics and Engineering* 395 (2022), p. 115019. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115019 (cit. on pp. 12, 190).

[69] Abhinav Muta, Prabhu Ramachandran, and Pawan Negi. “An efficient, open source, iterative ISPH scheme”. In: *Computer Physics Communications* 255 (2020), p. 107283. ISSN: 0010-4655 (cit. on pp. 11, 54, 190).

[70] Abhinav Muta, Prabhu Ramachandran, and Pawan Negi. “An efficient, open source, iterative ISPH scheme”. In: *Computer Physics Communications* 255 (2020), p. 107283. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2020.107283 (cit. on p. 28).

[71] Prapanch Nair and Gaurav Tomar. "Volume conservation issues in incompressible smoothed particle hydrodynamics". In: *Journal of Computational Physics* 297 (2015), pp. 689–699. ISSN: 0021-9991 (cit. on p. 26).

[72] Mark O. Neal and Ted Belytschko. "Explicit-explicit subcycling with non-integer time step ratios for structural dynamic systems". In: *Computers & Structures* 31.6 (1989), pp. 871–880. ISSN: 0045-7949 (cit. on p. 167).

[73] Pawan Negi and Prabhu Ramachandran. "A family of second order convergent weakly-compressible SPH schemes". In: *arXiv e-prints*, arXiv:2107.11859 (July 2021), arXiv:2107.11859. arXiv: 2107.11859 [math.NA] (cit. on p. 123).

[74] Pawan Negi and Prabhu Ramachandran. "A new family of second order convergent weakly-compressible SPH schemes". In: *arXiv e-prints*, arXiv:2107.11859 (July 2021), arXiv:2107.11859. arXiv: 2107.11859 [math.NA] (cit. on p. 167).

[75] Pawan Negi and Prabhu Ramachandran. "Algorithms for uniform particle initialization in domains with complex boundaries". In: *Computer Physics Communications* 265 (Aug. 2021), p. 108008. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2021.108008 (cit. on p. 117).

[76] Pawan Negi and Prabhu Ramachandran. "How to Train Your Solver: A Method of Manufactured Solutions for Weakly Compressible Smoothed Particle Hydrodynamics". In: *Physics of Fluids* 33.12 (Dec. 2021), p. 127108. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0072383 (cit. on p. 124).

[77] Pawan Negi and Prabhu Ramachandran. "How to train your solver: A method of manufactured solutions for weakly-compressible SPH". In: *arXiv e-prints*, arXiv:2109.09697 (Sept. 2021), arXiv:2109.09697. arXiv: 2109.09697 [physics.flu-dyn] (cit. on p. 167).

[78] Pawan Negi, Prabhu Ramachandran, and Asmelash Haftu. "An improved non-reflecting outlet boundary condition for weakly-compressible SPH". In: *ArXiV* (2019) (cit. on pp. 21, 33, 51, 52).

[79] Nomeritae et al. "Explicit incompressible SPH algorithm for free-surface flow modelling: A comparison with weakly compressible schemes". In: *Advances in Water Resources* 97 (2016), pp. 156–167. ISSN: 0309-1708 (cit. on pp. 3, 22, 27, 29, 40).

[80] G. Oger et al. “SPH accuracy improvement through the combination of a quasi-Lagr angian shifting transport velocity and consistent ALE formalisms”. In: *Journal of Computational Physics* 313 (May 2016), pp. 76–98. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.02.039 (cit. on p. 99).

[81] G. Oger et al. “SPH accuracy improvement through the combination of a quasi-Lagrangian shifting transport velocity and consistent ALE formalisms”. In: *Journal of Computational Physics* 313 (May 2016), pp. 76–98 (cit. on p. 19).

[82] OpenFOAM. *OpenFOAM: User Guide v2112*. URL: https://www.openfoam.com/documentation/guides/latest/doc/guide-overset.html (visited on 06/26/2023) (cit. on p. 160).

[83] Chong Peng et al. “A fully resolved SPH-DEM method for heterogeneous suspensions with arbitrary particle shape”. In: *Powder Technology* 387 (July 2021), pp. 509–526 (cit. on p. 9).

[84] Roger D. Peng. “Reproducible Research in Computational Science”. In: *Science* 334.6060 (2011), pp. 1226–1227. DOI: 10.1126/science.1213847 (cit. on p. 11).

[85] Ashkan Rafiee and Krish P. Thiagarajan. “An SPH projection method for simulating fluid-hypoelastic structure interaction”. In: *Computer Methods in Applied Mechanics and Engineering* 198.33 (2009), pp. 2785–2795. ISSN: 0045-7825 (cit. on pp. 3, 40).

[86] Prabhu Ramachandran. “Development and Study of a High-Resolution Two-Dimensional Random Vortex Method”. PhD thesis. Madras: IIT Madras, June 2004 (cit. on pp. 120, 132, 133, 135, 136, 138, 139, 147, 148, 166).

[87] Prabhu Ramachandran. “automan: A Python-Based Automation Framework for Numerical Computing”. In: *Computing in Science & Engineering* 20.5 (Sept. 2018), pp. 81–97 (cit. on pp. 11, 23, 56, 68, 94).

[88] Prabhu Ramachandran, Abhinav Muta, and M. Ramakrishna. “Dual-time smoothed particle hydrodynamics for incompressible fluid simulation”. In: *Computers & Fluids* 227 (2021), p. 105031. ISSN: 0045-7930 (cit. on pp. 12, 45, 166, 190).

[89] Prabhu Ramachandran and Kunal Puri. “Entropically damped artificial compressibility for SPH”. In: *Computers and Fluids* 179.30 (Jan. 2019), pp. 579–594. DOI: 10.1016/j.compfluid.2018.11.023 (cit. on pp. 4, 34, 52, 56, 57, 62, 68, 69, 91, 93, 97, 151, 165).

[90] Prabhu Ramachandran et al. “PySPH: A Python-Based Framework for Smoothed Particle Hydrodynamics”. In: *ACM Trans. Math. Softw.* 47.4 (Sept. 2021). ISSN: 0098-3500 (cit. on pp. 23, 48, 56, 68, 94, 157, 169, 170, 190).

[91] Prabhu Ramachandran et al. “PySPH: A Python-based Framework for Smoothed Particle Hydrodynamics”. In: *ACM Transactions on Mathematical Software* 47.4 (Dec. 2021), pp. 1–38. DOI: 10.1145/3460773 (cit. on p. 21).

[92] Stefan Reinhardt et al. “Fully Asynchronous SPH Simulation”. In: *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. Ed. by Bernhard Thomaszewski, KangKang Yin, and Rahul Narain. ACM, 2017. ISBN: 978-1-4503-5091-4. DOI: 10.1145/3099564.3099571 (cit. on p. 167).

[93] Yaidel Reyes López, Dirk Roose, and Carlos Recarey Morfa. “Dynamic Particle Refinement in SPH: Application to Free Surface Flow and Non-Cohesive Soil Simulations”. In: *Computational Mechanics* 51.5 (May 2013), pp. 731–741. ISSN: 0178-7675, 1432-0924. DOI: 10.1007/s00466-012-0748-0 (cit. on p. 5).

[94] Massoud Rezavand, Mohammad Taeibi-Rahni, and Wolfgang Rauch. “An ISPH Scheme for Numerical Simulation of Multiphase Flows with Complex Interfaces and High Density Ratios”. In: *Computers & Mathematics with Applications* 75.8 (Apr. 2018), pp. 2658–2677. ISSN: 08981221. DOI: 10.1016/j.camwa.2017.12.034 (cit. on p. 81).

[95] Daniel M. Robb, Susan J. Gaskin, and Jean-Christophe Marongiu. “SPH-DEM model for free-surface flows containing solids applied to river ice jams”. In: *Journal of Hydraulic Research* 54.1 (Jan. 2016), pp. 27–40 (cit. on p. 9).

[96] Martin Robinson, Paul Cleary, and Joseph Monaghan. “Analysis of mixing in a Twin Cam mixer using smoothed particle hydrodynamics”. In: *AIChE Journal* 54.8 (Aug. 2008), pp. 1987–1998 (cit. on p. 9).

[97] Emanuele Rossi et al. “The Diffused Vortex Hydrodynamics Method”. In: *Communications in Computational Physics* 18.2 (2015), pp. 351–379. DOI: 10.4208/cicp.271014.200415a (cit. on pp. 135, 140–142, 159).

[98] Louis F. Rossi. “Resurrecting Core Spreading Vortex Methods: A New Scheme that is Both Deterministic and Convergent”. In: *SIAM Journal on Scientific Computing* 17.2 (Mar. 1996). Publisher: Society for Industrial and Applied Mathematics, pp. 370–397. ISSN: 1064-8275 (cit. on p. 92).

[99] Stephan Rosswog. “Astrophysical smooth particle hydrodynamics”. In: *New Astronomy Reviews* 53.4 (2009), pp. 78–104. ISSN: 1387-6473 (cit. on p. 17).

[100] Stephan Rosswog. “SPH Methods in the Modelling of Compact Objects”. In: *Living Reviews in Computational Astrophysics* 1.1 (Dec. 2015), p. 1. ISSN: 2367-3621, 2365-0524. DOI: 10.1007/lrca-2015-1 (cit. on pp. 127, 128).

[101] Fardin Rouzbahani and Kazem Hejranfar. “A truly incompressible smoothed particle hydrodynamics based on artificial compressibility method”. In: *Computer Physics Communications* 210 (2017), pp. 10–28. ISSN: 0010-4655 (cit. on pp. 4, 55–58, 68, 91).

[102] Songdong Shao and Edmond Y.M. Lo. “Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface”. In: *Advances in Water Resources* 26.7 (2003), pp. 787–800. ISSN: 0309-1708 (cit. on p. 3).

[103] Alex Skillen et al. “Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised Fickian smoothing applied to body–water slam and efficient wave–body interaction”. In: *Computer Methods in Applied Mechanics and Engineering* 265 (2013), pp. 163–173. ISSN: 0045-7825. DOI: 10.1016/j.cma.2013.05.017 (cit. on pp. 7, 24, 35).

[104] B. Solenthaler and R. Pajarola. “Predictive-corrective Incompressible SPH”. In: *ACM Transactions on Graphics* 28.3 (July 2009), 40:1–40:6. ISSN: 0730-0301 (cit. on p. 4).

[105] Barbara Solenthaler and Markus Gross. “Two-Scale Particle Simulation”. In: *ACM SIGGRAPH 2011 Papers on - SIGGRAPH ’11*. Vancouver, British Columbia, Canada: ACM Press, 2011, p. 1. ISBN: 978-1-4503-0943-1. DOI: 10.1145/1964921.1964976 (cit. on pp. 6, 92).

[106] *SPH rEsearch and engineeRing International Community*. Accessed: 2022-03-25. Mar. 2022 (cit. on p. 144).

[107] Fabian Spreng et al. “A Local Adaptive Discretization Algorithm for Smoothed Particle Hydrodynamics: For the Inaugural Issue”. In: *Computational Particle Mechanics* 1.2 (June 2014), pp. 131–145. ISSN: 2196-4378, 2196-4386. DOI: 10.1007/s40571-014-0015-6 (cit. on pp. 6, 9).

[108] Fabian Spreng et al. “An Advanced Study on Discretization-Error-Based Adaptivity in Smoothed Particle Hydrodynamics”. In: *Computers & Fluids* 198 (Feb. 2020), p. 104388. ISSN: 00457930. DOI: 10.1016/j.compfluid.2019.104388 (cit. on pp. 6, 93).

[109] Volker Springel. “E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh”. In: *Monthly Notices of the Royal Astronomical Society* 401.2 (Jan. 2010), pp. 791–851. DOI: 10.1111/j.1365-2966.2009.15715.x. arXiv: 0901.4107 [astro-ph.CO] (cit. on p. 167).

[110] Volker Springel. “Smoothed Particle Hydrodynamics in Astrophysics”. In: *Annual Review of Astronomy and Astrophysics* 48.1 (Aug. 2010), pp. 391–430. ISSN: 0066-4146, 1545-4282. DOI: 10.1146/annurev-astro-081309-130914 (cit. on p. 127).

[111] Shankar Subramaniam. “A new mesh-free vortex method”. ENGLISH. PhD thesis. The Florida State University, FAMU-FSU College of Engineering, 1996 (cit. on pp. 136, 138).

[112] P. N. Sun et al. “The δ+-SPH model: Simple procedures for a further improvement of the SPH scheme”. In: *Computer Methods in Applied Mechanics and Engineering* 315 (Mar. 2017), pp. 25–49. ISSN: 0045-7825 (cit. on pp. 5, 8, 99).

[113] P. N. Sun et al. “Multi-resolution Delta-plus-SPH with tensile instability control: Towards high Reynolds number flows”. In: *Computer Physics Communications* 224 (Mar. 2018), pp. 63–80. ISSN: 0010-4655 (cit. on pp. 5, 92, 94, 140–142, 159).

[114] P. N. Sun et al. “A consistent approach to particle shifting in the δ-Plus-SPH model”. In: *Computer Methods in Applied Mechanics and Engineering* 348 (May 2019), pp. 912–934. ISSN: 0045-7825. DOI: 10.1016/j.cma.2019.01.045 (cit. on pp. 98, 121, 122).

[115] P. N. Sun et al. “An Accurate SPH Volume Adaptive Scheme for Modeling Strongly-Compressible Multiphase Flows. Part 1: Numerical Scheme and Validations with Basic 1D and 2D Benchmarks”. In: *Journal of Computational Physics* 426 (Feb. 2021), p. 109937. ISSN: 00219991. DOI: 10.1016/j.jcp.2020.109937 (cit. on pp. 6, 93).

[116] P.N. Sun et al. “The δ^+-SPH Model: Simple Procedures for a Further Improvement of the SPH Scheme”. In: *Computer Methods in Applied Mechanics and Engineering* 315 (Mar. 2017), pp. 25–49. ISSN: 00457825. DOI: 10.1016/j.cma.2016.10.028 (cit. on p. 79).

[117] P.N. Sun et al. “A consistent approach to particle shifting in the δ-Plus-SPH model”. In: *Computer Methods in Applied Mechanics and Engineering* 348 (May 2019), pp. 912–934. DOI: 10.1016/j.cma.2019.01.045 (cit. on p. 19).

[118] A. Tafuni et al. “A versatile algorithm for the treatment of of open boundary conditions in smoothed particle hydrodynamics GPU models”. In: *Computer methods in applied mechanical engineering* 342 (2018), pp. 604–624. DOI: 10.1016/j.cma.2018.08.004 (cit. on p. 51).

[119] Hai Tan and Shenghong Chen. “A hybrid DEM-SPH model for deformable landslide and its generated surge waves”. In: *Advances in Water Resources* 108 (Oct. 2017), pp. 256–276 (cit. on p. 9).

[120] R Vacondio, BD Rogers, and PK Stansby. “Accurate particle splitting for smoothed particle hydrodynamics in shallow water with shock capturing”. In: *International Journal for Numerical Methods in Fluids* 69.8 (2012), pp. 1377–1410. DOI: 10.1002/fld.2646 (cit. on pp. 10, 94, 164).

[121] R Vacondio et al. “Variable resolution for SPH: a dynamic particle coalescing and splitting scheme”. In: *Computer Methods in Applied Mechanics and Engineering* 256 (Apr. 2013), pp. 132–148. DOI: 10.1016/j.cma.2012.12.014 (cit. on p. 144).

[122] R. Vacondio, B. D. Rogers, and P. K. Stansby. “Accurate particle splitting for smoothed particle hydrodynamics in shallow water with shock capturing”. In: *International Journal for Numerical Methods in Fluids* 69.8 (July 20, 2012), pp. 1377–1410. ISSN: 02712091 (cit. on pp. 5, 98, 104, 115, 116, 118–120, 151).

[123] R. Vacondio et al. “Variable resolution for SPH: A dynamic particle coalescing and splitting scheme”. In: *Computer Methods in Applied Mechanics and Engineering* 256 (Apr. 2013), pp. 132–148. ISSN: 0045-7825 (cit. on pp. 5, 6, 8, 92, 93, 103, 106, 111).

[124] R. Vacondio et al. “Variable resolution for SPH in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity”. In: *Computer Methods in Applied Mechanics and Engineering* 300 (Mar. 1, 2016), pp. 442–460. ISSN: 0045-7825 (cit. on pp. 5, 93).

[125] Renato Vacondio et al. “Grand challenges for Smoothed Particle Hydrodynamics numerical schemes”. In: *Computational Particle Mechanics* (Sept. 2020). ISSN: 2196-4386 (cit. on p. 92).

[126] Damien Violeau. *Fluid mechanics and the SPH method: theory and applications*. 1st ed. OCLC: ocn776772541. Oxford ; New York: Oxford University Press, 2012. 594 pp. ISBN: 978-0-19-965552-6 (cit. on p. 13).

[127] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 35).

[128] H. A. van der Vorst. “Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644. eprint: https://doi.org/10.1137/0913035 (cit. on p. 29).

[129] H. G. Weller et al. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in Physics* 12.6 (1998), p. 620 (cit. on p. 159).

[130] Y.S. Wu and P. Smolinski. “A multi-time step integration algorithm for structural dynamics based on the modified trapezoidal rule”. In: *Computer Methods in Applied Mechanics and Engineering* 187.3 (2000), pp. 641–660. ISSN: 0045-7825 (cit. on p. 167).

[131] Rui Xu, Peter Stansby, and Dominique Laurence. “Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach”. In: *Journal of Computational Physics* 228.18 (2009), pp. 6703–6725. DOI: 10.1016/j.jcp.2009.05.032 (cit. on pp. 7, 19, 24, 35, 36).

[132] Xiufeng Yang and Song-Charng Kong. “Smoothed particle hydrodynamics method for evaporating multiphase flows”. In: *Physical Review E* 96.3 (Sept. 2017), p. 033309. ISSN: 2470-0045, 2470-0053 (cit. on pp. 5, 9).

[133] Xiufeng Yang and Song-Charng Kong. “Adaptive resolution for multiphase smoothed particle hydrodynamics”. In: *Computer Physics Communications* 239 (June 2019), pp. 112–125. ISSN: 00104655 (cit. on pp. 5, 6, 93, 103, 111).

[134] Chi Zhang, Massoud Rezavand, and Xiangyu Hu. “Dual-criteria time stepping for weakly compressible smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 404 (Mar. 2020), p. 109135. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.109135 (cit. on pp. 4, 8, 56).

[135] Chi. Zhang, Xiangyu Y. T. Hu, and Nikolaus A Adams. “A generalized transport-velocity formulation for smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 337 (2017), pp. 216–232 (cit. on pp. 7, 20, 24, 25).

List of Publications

1. Asmelash Haftu, Abhinav Muta, and Prabhu Ramachandran. “Parallel Adaptive Weakly-Compressible SPH for Complex Moving Geometries”. In: *Computer Physics Communications* 277 (Aug. 2022), p. 108377. ISSN: 00104655. DOI: 10.1016/j.cpc.2022.108377

2. Abhinav Muta and Prabhu Ramachandran. “Efficient and accurate adaptive resolution for weakly-compressible SPH”. in: *Computer Methods in Applied Mechanics and Engineering* 395 (2022), p. 115019. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115019

3. Prabhu Ramachandran, Abhinav Muta, and M. Ramakrishna. “Dual-time smoothed particle hydrodynamics for incompressible fluid simulation”. In: *Computers & Fluids* 227 (2021), p. 105031. ISSN: 0045-7930

4. Abhinav Muta, Prabhu Ramachandran, and Pawan Negi. “An efficient, open source, iterative ISPH scheme”. In: *Computer Physics Communications* 255 (2020), p. 107283. ISSN: 0010-4655

5. Prabhu Ramachandran, Aditya Bhosale, Kunal Puri, Pawan Negi, Abhinav Muta, A. Dinesh, Dileep Menon, Rahul Govind, Suraj Sanka, Amal S. Sebastian, Ananyo Sen, Rohan Kaushik, Anshuman Kumar, Vikas Kurapati, Mrinalgouda Patil, Deep Tavker, Pankaj Pandey, Chandrashekhar Kaushik, Arkopal Dutt, and Arpit Agarwal. “PySPH: A Python-Based Framework for Smoothed Particle Hydrodynamics”. In: *ACM Trans. Math. Softw.* 47.4 (Sept. 2021). ISSN: 0098-3500