

Development of Second-Order Convergent Weakly-Compressible Smoothed Particle Hydrodynamics Schemes

A Thesis
Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy
by

Pawan Singh Negi



Department of Aerospace Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

17 May 2023

Dedicated to my loving parents and wife

Approval Sheet

This thesis entitled “Development of Second-Order Convergent Weakly-Compressible Smoothed Particle Hydrodynamics Schemes” by Pawan Singh Negi is approved for the degree of Doctor of Philosophy.



Prof. Amithabh Bhattacharya



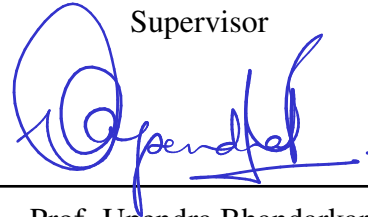
Prof. Aniruddha Sinha

Examiners



Prof. Prabhu Ramachandran

Supervisor



Prof. Upendra Bhandarkar

Chairman

Date: 17 May 2023

Place: Mumbai

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 declare that I have properly and accurately acknowledged all sources used in the production of this report. 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 a 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.

Date: 17 May 2023



Pawan Singh Negi

Abstract

The convergence of a Smoothed Particle Hydrodynamics (SPH) scheme is a grand challenge problem. In SPH, the accuracy (and sometimes the rate of convergence) is shown by comparing the solution with established solutions or using exact solutions. However, the convergence achieved is poor. In this work, we propose techniques to construct a Lagrangian, second-order convergent, weakly-compressible SPH (WCSPH) scheme. We systematically study various aspects of SPH approximation, viz. type of kernel, smoothing radius, discretization formulation, and kernel gradient correction to identify the root cause of the problem of lack of convergence. We compare various discretization formulations proposed in the SPH literature and identify operators that show convergence. We then construct a scheme using convergent discretizations and compare it with the existing schemes in the literature by solving the Taylor-Green problem. We then investigate the effect of the speed of sound on the convergence of the scheme and recommend the criteria to set the artificial speed of sound. We then solve inviscid problems like the Gresho-Chan vortex and the incompressible shear layer to demonstrate the conservation properties of the newly proposed and existing schemes. We propose variations of the newly proposed scheme such that they use the transport velocity or an Eulerian formulation while still showing convergence.

Subsequently, we focus on the efficiency of the entire process to obtain the order of convergence. This process is a part of the verification of a code. The method of exact solutions and comparison with existing solutions requires a lot of computational time. Furthermore, if the solution diverges, these methods do not give insight into specific problems in the code. Moreover, these methods are inconclusive while comparing the convergence of boundary condition implementations. We propose the method of manufactured solutions (MMS) for SPH. For the first time, we use MMS in the context of a general Lagrangian WCSPH code. We propose a set of instructions to construct manufactured solutions to verify Lagrangian implementations. We demonstrate the use of MMS to obtain the convergence of all the proposed schemes. The MMS requires much less computation time than the time taken by the state-of-the-art techniques employed in the SPH literature.

We also demonstrate the use of MMS to obtain convergence for extremely high-resolution and three-dimensional problems, which is very difficult with traditional methods.

We extend the method to verify the convergence of solid wall and inflow/outflow boundary condition implementations. However, in the presence of an arbitrarily-shaped solid, the particles placed on a Cartesian grid do not capture the features of the geometry properly. We propose a novel particle packing technique that simultaneously arranges the solid and fluid particles around the solid boundary surface such that the geometric features are captured while the particle density is uniform. We use this packing technique to generate test cases to verify solid boundary conditions.

We propose various manufactured solutions for the Neumann pressure, slip, and no-slip boundary conditions for different shapes of the solid boundary, viz. straight, concave, and convex. We also propose different MSs for inflow and outflow boundary conditions. We use these MSs to verify, for the first time, the convergence of various existing boundary condition implementations in SPH literature. In the case of inflow and outflow, we propose modified implementations that show convergence in the presence of a wave traveling across the fluid-outlet interface. We finally put together all the second-order convergent discretizations and boundary implementations and propose a complete second-order convergent scheme. This new scheme can simulate a wind tunnel-like simulation with an arbitrarily-shaped solid downstream of the flow with high accuracy. We demonstrate the accuracy achieved by simulating the flow past a circular cylinder.

Table of contents

Abstract	ix
List of figures	xv
List of tables	xxv
Nomenclature	xxix
1 Introduction	1
1.1 The SPH method	1
1.2 Discrete SPH approximation	3
1.3 SPH discretization of operators	5
1.3.1 Divergence discretization	6
1.3.2 Gradient discretization	6
1.3.3 Laplacian discretization	8
1.4 Incompressible fluid flow methods in SPH	10
1.4.1 Incompressible SPH	10
1.4.2 Weakly-compressible SPH	10
1.5 Verification of an implementation	11
1.5.1 Literature survey	12
1.6 Objective of current work	14
1.7 Outline	15
2 Construction of second-order convergent WCSPH schemes	17
2.1 Selection of approximating kernel	18
2.1.1 Unperturbed periodic domain	22
2.1.2 Perturbed periodic domain	24
2.2 Comparison of discretization operators	25
2.2.1 Comparison of $\nabla \cdot \mathbf{u}$ approximation	25
2.2.2 Comparison of $\frac{\nabla p}{\rho}$ approximation	27

2.2.3	Comparison of $\nabla^2 \mathbf{u}$ approximation	30
2.3	The second-order convergent WCSPH scheme	33
2.3.1	Considerations while applying kernel gradient correction	33
2.3.2	Considerations for the initial particle distribution	34
2.3.3	Minimal requirements for a SOC scheme	35
2.3.4	The effect of c_o on convergence	38
2.3.5	Variations of the SOC scheme	38
2.4	Results and discussions	41
2.4.1	Comparison with existing SPH schemes	41
2.4.2	Convergence with varying speed of sound	44
2.4.3	Comparison of SOC variants	46
2.4.4	Comparison of conservation errors	48
2.4.5	Long time simulations	52
2.4.6	Cost of computation	53
2.5	Summary	55
3	Verification techniques for WCSPH schemes	59
3.1	Drawback of current verification methods	61
3.2	The method of manufactured solutions	63
3.2.1	Implementation of MMS	65
3.3	Application of the MMS	69
3.3.1	The effect of initial particle configuration	70
3.3.2	The selection of the domain shape	74
3.3.3	Comparison of EDAC and PE-IPST-C	75
3.3.4	Comparison of E-C and TV-C	77
3.3.5	Identification of mistakes in the implementation	78
3.3.6	Convergence at extreme resolutions	81
3.3.7	Verification in 3D	85
3.4	Summary	85
4	Construction of solid bodies in SPH	89
4.1	Particle packing algorithms	89
4.1.1	Standard packing	90
4.1.2	Coupled packing	91
4.1.3	Hybrid packing	94
4.2	Comparison of particle packing algorithms	109
4.3	Summary	117

5	Boundary condition implementations in SPH	119
5.1	Solid boundary conditions	119
5.2	Open boundary conditions	126
5.2.1	Test cases for open BC implementations	130
5.3	Summary	144
6	Verification of boundary condition implementations	145
6.1	Manufactured solutions for BCs	146
6.1.1	Neumann pressure boundary	148
6.1.2	Slip boundary condition	149
6.1.3	No-slip boundary condition	151
6.1.4	Inlet and outlet velocity boundary condition	153
6.1.5	Inlet and outlet pressure boundary condition	155
6.2	Verification of BC implementations	157
6.2.1	Comparison of solid BC implementations	157
6.2.2	Comparison of open BC implementations	168
6.3	Performance comparison	174
6.4	Complete second-order convergent SPH scheme	174
6.5	Summary	180
7	Conclusions and future work	181
7.1	Future work	183
A	Supporting material	185
A.1	Skew-adjoint pressure gradient and velocity divergence operators	185
A.2	Instabilities inherent to the smoothing kernel	186
A.3	Derivation of kernel gradient correction	186
A.3.1	Correction by Bonet et al. (1999)	186
A.3.2	Correction by Liu et al. (2006)	187
A.3.3	Correction by Huang et al. (2019)	188
A.3.4	Correction by Fatehi et al. (2011)	188
A.3.5	Correction by Korzilius et al. (2017)	190
A.4	Derivation of error terms in SPH approximations	191
A.4.1	Error in continuous SPH interpolation	192
A.4.2	Error in continuous SPH gradient interpolation	193
A.4.3	Error estimation of discrete SPH gradient approximation	193
A.4.4	Error estimation of discrete SPH Laplacian approximation	195

A.4.5	Quadrature error in discrete SPH gradient approximation	198
A.4.6	Quadrature error in discrete SPH gradient approximation on a non-uniform particle distribution	201
A.5	Weakly-compressible SPH schemes	204
A.5.1	Standard scheme	204
A.5.2	δ -SPH scheme	204
A.5.3	Transport Velocity Formulation (TVF)	205
A.5.4	Entropically Damped Artificial Compressibility (EDAC) scheme .	206
A.5.5	Arbitrary Lagrange Eulerian SPH (ALE-SPH) and δ^+ SPH scheme	206
A.5.6	Eulerian WCSPH scheme	207
A.6	δ^+ SPH formulation correction	208
A.7	Schemes with issues solving the Gresho-Chan vortex	209
List of publications		225
Acknowledgements		227

List of figures

1.1	Plots of some of the widely used SPH kernels.	3
1.2	Example of a discretized domain with a kernel.	4
2.1	The unperturbed periodic particle and perturbed periodic particles.	18
2.2	The particle density for different kernels with varying numbers of neighbors. 19	
2.3	Order of convergence for the approximation of a function for different $h_{\Delta x}$ values in a UP domain. The dashed line shows the second-order rate.	20
2.4	Order of convergence for derivative approximation for different $h_{\Delta x}$ values in a UP domain. The dashed line shows the second-order rate.	21
2.5	Order of convergence for function approximation for different $h_{\Delta x}$ values in a PP domain. The dashed line shows the second-order rate.	22
2.6	Order of convergence for the derivative approximation for different $h_{\Delta x}$ values in a PP domain. The dashed line shows the second-order rate. Equation (1.23) is used for the approximation.	23
2.7	The convergence of the derivative approximation with different kernels when the kernel gradient correction of Bonet et al. (1999) is used on a PP domain. We use $h_{\Delta x} = 1.2$	25
2.8	The rate of convergence in UP (left) and PP (right) domains for velocity divergence approximation in eq. (2.4). The dashed line shows the SOC rate. The suffix <code>_bc</code> represents the corresponding form with Bonet correction.	26
2.9	The rate of convergence in UP (left) and PP (right) domains for divergence approximation of a divergence-free velocity field. The dashed line shows the SOC rate. The suffix <code>_bc</code> represents the corresponding form with Bonet correction.	27

2.10	The rate of convergence in UP (left) and PP (right) domains for various pressure gradient listed in table 2.2. The dashed line shows the SOC rate. The <code>_bc</code> and <code>_lc</code> suffixes represent the corresponding form with Bonet correction and Liu correction, respectively. The <code>sym_sl</code> is the <code>sym1</code> formulation with symmetrization of kernel proposed in Dilts (1999).	28
2.11	The rate of convergence UP (left) and PP (right) domains for various approximations of the Laplacian operator in table 2.4. The dashed line shows the SOC rate. The suffixes <code>_bc</code> and <code>_lc</code> represent the corresponding form with the Bonet correction and Liu correction, respectively. The <code>Fatehi_c</code> refers to the <code>fatehi</code> formulation with the correction proposed by Fatehi et al. (2011) (see appendix A.3.4).	31
2.12	Convergence of L_1 error in pressure (left) and velocity (right) with the change in resolution. $Re = 100$, $c_o = 10m/s$, $\Delta t = 6.54 \times 10^{-5}s$ and only 1 timestep taken.	43
2.13	Convergence rates for pressure (left) and velocity (right). The L-IPST-C and L-IPST-F methods are compared for different values of c_o	45
2.14	Convergence rates for pressure (left) and velocity (right) of different variants of the SOC scheme.	46
2.15	The velocity of particles with the distance from the center of the vortex (left) and the x -component of the total linear momentum (right) for existing and L-IPST-C schemes.	49
2.16	The velocity of particles with the distance from the center of the vortex (left) and the x -component of the total linear momentum (right) for the variation of the SOC scheme.	49
2.17	The angular momentum variation with time for Gresho-Chan vortex for different schemes.	50
2.18	Vorticity contour plot for 500×500 resolution for existing and L-IPST-C schemes.	51
2.19	Vorticity contour plot for 500×500 resolution for all the variation of the SOC scheme.	52
2.20	The maximum velocity decay (left) and kinetic energy of the flow with respect time (right) for the Taylor-Green problem.	53
2.21	The velocity of all the particles in the domain with distance from the center (top left), linear (top right) and angular (bottom) momentum with respect to time for Gresho-Chan vortex problem.	54

2.22	The L_1 error in velocity with respect to the time taken to evaluate 5000 timesteps for all the schemes discussed in the previous sections.	55
3.1	The decay in velocity magnitude for different resolutions compared with the exact solution for the Taylor-Green problem.	61
3.2	The velocity along x and y directions along the center line $x = 0.5m$ of the domain for the lid-driven cavity problem.	62
3.3	The L_1 error in velocity for the Taylor-Green problem.	63
3.4	The different initial particle arrangements in blue with the solid boundary in orange.	70
3.5	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.7) and the source term in eq. (3.8) after 10 timesteps for different configurations.	72
3.6	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.7) and the source term in eq. (3.8) after 100 timesteps for all the configurations.	72
3.7	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.9) and the corresponding source terms after 100 timesteps for all the configurations.	73
3.8	The different domain shapes with solid particles in orange and fluid particles in blue.	74
3.9	The L_1 error in pressure (left) and velocity (right) with increase in resolution for different shapes of the domain.	74
3.10	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.10), and the source term in eq. (3.11) for EDAC and eq. (3.12) for PE-IPST-C after one timestep.	76
3.11	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.10), and the source term in eq. (3.11) for EDAC and eq. (3.12) for PE-IPST-C after 100 timesteps.	76
3.12	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.10) and the source term in eq. (3.14) after 100 timesteps for the different schemes.	78
3.13	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.16) and the corresponding source term after 1 timestep for L-IPST-C and the scheme where the divergence is computed using the incorrect eq. (3.15).	79

3.14	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.16) and the corresponding source term after one timestep for L-IPST-C and the scheme where the pressure gradient is computed using symmetric formulation.	80
3.15	The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.9) and the corresponding source term after one timestep for L-IPST-C and the scheme with the viscous term discretized using formulation given by Cleary et al. (1999).	81
3.16	The domain filled by blue fluid particles. The red box shows the smallest domain considered for the highest resolution of 8000×8000 and the black box shows the area which is considered to evaluate error for all the resolutions.	82
3.17	The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.9). All cases are solved using the L-IPST-C scheme with kernel gradient correction.	83
3.18	The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.9). All cases are solved using the L-IPST-C scheme with no kernel gradient correction.	83
3.19	The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.18). All cases are solved using the L-IPST-C scheme with kernel gradient correction.	84
3.20	The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.18). All cases are solved using the L-IPST-C scheme with no kernel gradient correction.	84
3.21	The L_{∞} error in pressure (left) and velocity (right) after 10 timesteps as a function of resolution solved using L-IPST-C scheme with and without kernel gradient correction. The source term is calculated using the MS in eq. (3.19).	86
4.1	Schematic for the Coupled packing algorithm. The external region is marked as domain 2, and the internal region is marked as domain 1.	93
4.2	Schematic of the initial distribution of particles and the different kinds of particles.	95
4.3	Flowchart of the particle packing algorithm. The box outlined in dashed red lines is the initial projection phase.	96

4.4	The pre-processed patch (within dashed lines) of particles placed in the appropriate location of a typical simulation. The blue particles denote the free particles identified as fluid particles, and the red represents the solid particles identified by the packing process. The green particles are generated from a fixed mesh of points.	97
4.5	Shifting of the boundary near a sharp-edged boundary. The boundary nodes are depicted in red with normals. On the left is the boundary surface after the initial shifting shown as black dashed lines. On the right is the final geometry after the removal of the intersecting edges, which are shown as dashed red lines. The annular blue free particle is the candidate to be placed on the corner and held fixed.	99
4.6	Force due to LJ potential and the gradient of eq. (4.12) as a function of distance, r and $\alpha = 1$	100
4.7	Different packing structures in 2D.	102
4.8	Particle density convergence with a particle at center perturbed by $\Delta x/4$	103
4.9	Error in $\nabla\psi_i$ for a domain with unit radius cylinder with different projection frequencies.	104
4.10	Motion of boundary particle along the geometry.	106
4.11	Solid (red) and fluid (blue) particles for a circular cylinder for $\Delta x = 0.1$	110
4.12	Particle density distribution of the packed particles for the circular cylinder geometry for $\Delta x = 0.1$	111
4.13	L_1 error for SPH approximation of function and its derivative for the circular cylinder geometry.	111
4.14	Solids (red) and fluids (blue) for the zig-zag wall for $\Delta x = 0.05$	112
4.15	Particle density distribution for the zig-zag wall for $\Delta x = 0.05$	113
4.16	L_1 error for SPH approximation of function and its derivative for the zig-zag wall.	113
4.17	Particle density distribution at different resolutions for an arbitrarily-shaped object.	115
4.18	The density distribution for a symmetric airfoil with particle spacing $\Delta x = 0.02$ and convergence tolerance $\epsilon = 2.5 \times 10^{-5}$	116
4.19	Particle density distribution on the surface of the ellipsoid for $\Delta x = 0.1$	116
4.20	L_1 error for SPH approximation of function and its derivative for the ellipsoid.	117
4.21	Different views showing particles (colored with particle density values) over the Stanford bunny.	118

5.1	L_1 error in pressure gradient (left) and Laplacian (right) approximation with change in the skipped number of layers.	120
5.2	Arrangement of ghost and virtual particles for a given fluid particle denoted by blue circles. The dashed line is the boundary surface, red particles are ghost (solid) particles. The black dots are generated to evaluate the gradient using finite differences at the red boundary points.	122
5.3	Arrangement of ghost and virtual particle for given blue fluid particles. The dashed line shows the solid boundary. The red circles on the right are the ghost particles, and the red crosses are created by reflecting the ghost about the interface.	123
5.4	Arrangement of particles for the method proposed by Takeda et al. (1994). The ghost particles are shown in red, and the fluid particles are shown in blue color. The value of the red particles are interpolated using the nearest fluid particle along the normal of the boundary surface.	124
5.5	Arrangement of particles for the method proposed by Randles et al. (1996) and Adami et al. (2012) for the fluid particles in blue. The boundary interface is shown by the dashed line. The red circles on the left represent the ghost solid particles. The property of the light blue fluid particles are manipulated to satisfy boundary conditions.	125
5.6	Arrangement of particles for the method proposed by Colagrossi et al. (2003) for the fluid particles in blue, and boundary represented by the dashed line. The ghost particles in red represent the solid created by the mirror reflection of fluid particles about the interface.	125
5.7	Schematic of the arrangement of fluid with the inlet and outlet particles. The top and bottom are supported by solid particles not shown in the figure.	126
5.8	Schematic of the arrangement of fluid with the inlet and outlet particles and their corresponding virtual particles are shown in light red and red, respectively.	128
5.9	Sketch of the domain used for backward-facing step simulations (all dimensions in mm).	131
5.10	Velocity at $t = 1.2sec$ for $Re = 389$ at different locations.	132
5.11	Sketch of the domain used for the flow past a circular cylinder simulations.	133
5.12	Normalized pressure at $t = 150sec$ for $Re = 200$	134
5.13	Normalized velocity at $t = 150sec$ for $Re = 200$	135
5.14	Plot for c_d for all methods at $Re=200$	136

5.15	Pressure plot at various times for the different methods. The solid line denotes the solution with the long domain.	137
5.16	Normalized pressure (left) and velocity (right) plots at $x = 1.7m$ with time for 2D varying inlet.	138
5.17	Normalized pressure (left) and velocity (right) along the $y = 0$ line for the 2D pulse problem. The left of the dashed red line is fluid, and the right is outlet region.	139
5.18	Normalized pressure (left) and velocity gradients (right) along the $y = 0$ line for the 2D pulse problem. Left of the dashed red line is fluid and right is outlet region.	139
5.19	Normalized pressure (left) and velocity (right) plots at $x = 1.7m$ with time for ramp inlet.	141
5.20	Normalized pressure (left), u-velocity (right) and v-velocity (center) plots at $x = 1.7m$ with time for 2D vortex advection with $1m/s$	142
6.1	Types of domains considered to test the convergence of solid boundary implementation. The fluid particles are represented by the blue color, the particles in green represent ghost particles on which properties are set using MS, and the particles in orange are used to test the convergence of boundary implementation of interest.	146
6.2	The packed version of convex and concave domains. The fluid particles are represented by blue color, the particles in green represents ghost particles on which properties are set using MS, and the particles in orange are used to test the convergence of desired boundary condition implementation.	147
6.3	The domain used for the verification of inlet and outlet boundary implementation. The blue particles represent the fluid, orange particle represent the wall, green particles are inflow particles, and red particles are the out-flow particles.	147
6.4	Velocity and pressure contours on the straight domain in fig. 6.1 of the MS in eq. (6.2).	148
6.5	Velocity and pressure contours on the convex/concave domain of the MS in eq. (6.3).	149
6.6	Velocity and pressure contours on the straight domain of the MS in eq. (6.4).	150
6.7	Velocity and pressure contours on the convex/concave domain of the MS in eq. (6.5).	150
6.8	Velocity and pressure contours on the straight domain of the MS in eq. (6.6).	151
6.9	Velocity and pressure contours on the convex domain of the MS in eq. (6.7).	152

6.10	Velocity and pressure contours on the concave domain of the MS in eq. (6.8).	152
6.11	Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.9).	153
6.12	Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.10).	154
6.13	Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.11).	154
6.14	Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.12).	155
6.15	Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.13).	156
6.16	Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.14).	156
6.17	L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the straight domain in fig. 6.1.	158
6.18	L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the convex domain in fig. 6.1.	158
6.19	L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the concave domain in fig. 6.1.	159
6.20	L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the packed-convex domain in fig. 6.2.	160
6.21	L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the packed-concave domain in fig. 6.2.	160
6.22	L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in straight domain in fig. 6.1.	161
6.23	L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the convex domain in fig. 6.1.	162
6.24	L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the concave domain in fig. 6.1.	162
6.25	L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the packed-convex domain in fig. 6.2.	163
6.26	L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the packed-concave domain in fig. 6.2.	163
6.27	L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the straight domain in fig. 6.1.	164

6.28	L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the convex domain in fig. 6.1.	165
6.29	L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the concave domain in fig. 6.1.	165
6.30	L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the packed-convex domain in fig. 6.2. Note that the Marrone plot overlaps the plot of MMS.	166
6.31	L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the packed-concave domain in fig. 6.2. Note that Marrone results overlaps the results of MMS.	167
6.32	L_1 error in pressure and velocity after 100 time steps for different inlet velocity boundary implementations in the domain in fig. 6.3.	169
6.33	L_1 error in pressure and velocity after 100 time steps for different inlet pressure boundary implementations in the domain in fig. 6.3.	169
6.34	L_1 error in pressure and velocity after 500 time steps for different inlet velocity wave going upstream boundary implementations in the domain in fig. 6.3.	170
6.35	L_1 error in pressure and velocity after 500 time steps for different inlet pressure wave going upstream boundary implementations in the domain in fig. 6.3.	170
6.36	L_1 error in pressure and velocity after 100 time steps for different outlet velocity boundary implementations in the domain in fig. 6.3.	171
6.37	L_1 error in pressure and velocity after 100 time steps for different outlet pressure boundary implementations in the domain in fig. 6.3.	172
6.38	L_1 error in pressure and velocity after 500 time steps for different outlet velocity wave going downstream boundary implementations in the domain in fig. 6.3.	172
6.39	L_1 error in pressure and velocity after 500 time steps for different outlet pressure boundary wave going downstream implementations in the domain in fig. 6.3.	173
6.40	The time taken with the increase in the number of threads for different solid boundary conditions.	175
6.41	Description of the domain of the flow past cylinder problem.	177
6.42	Average pressure at $t = 100sec$	178
6.43	Velocity magnitude at $t = 100sec$	179
6.44	c_d and c_l variation for the flow past a cylinder with time.	179

- A.1 The velocity of particles with the distance from the center of the vortex (left) and the x -component of the total linear momentum (right) for the Gresho-Chan vortex problem. 210

List of tables

2.1	Kernels and their properties (β reported for 1D kernels)	19
2.2	Different gradient approximations for $\frac{\nabla p}{\rho}$. The formulations with blank “Used in” columns are not used in any existing scheme.	28
2.3	The ratio $\frac{F_T}{F_{max}}$ showing the total force in the system due to the lack of conservation in the approximation, the time taken T_r relative to the asym formulation, the L_1 error for 500×500 particle in a PP domain, and the last column shows the order of convergence for all the methods listed in the first column. The formulations that show convergence is shown in red.	29
2.4	The various approximations of $\nabla^2 \mathbf{u}$. The formulations with blank “Used in” columns are not used in any existing scheme.	31
2.5	The ratio $\frac{F_T}{F_{max}}$ showing the total force in the system due to the lack of conservation of the approximation and the time taken, T_r relative to the tvf formulation, and L_1 error for 500×500 particle case in a PP domain. The last column shows the order of convergence for all the methods listed in the first column. The formulations that show convergence is shown in red.	32
2.6	The operators and their discretization suitable for a SOC scheme (For details, refer section 2.2).	36
2.7	Table showing total force w.r.t. the maximum force in the domain and the time taken for 1 iteration w.r.t. the TVF scheme for all the schemes. The last two columns show L_1 error in velocity and pressure at 500×500 resolution with the order of convergence in the brackets.	44
2.8	Comparison of the total force, time taken relative to the L-IPST-C with $c_o = 20m/s$, L_1 error in velocity and pressure at 500×500 resolution with order of convergence in the brackets for different values of c_o	46
2.9	Comparison of the total force, time taken relative to L-IPST-C, L_1 error in velocity and pressure at 500×500 resolution with order of convergence in brackets for variation of SOC scheme with $c_o = 80m/s$	47

4.1	The table shows the effect of the change in tolerance for convergence on the number of iterations and the L_∞ error in the density.	114
5.1	The reattachment length for $Re = 389$ for different outlet implementations.	131
5.2	Comparison of c_l , c_d and St values for different methods for $Re = 200$. . .	135
5.3	L_2 error in the p^* and u^* measured at the probe for the 2D pulse problem.	138
5.4	L_2 error in p^* measured at the probe for the 2D pulse with the change in EDAC parameter α	140
5.5	L_2 error in the p^* and u^* measured at the probe for the ramp velocity problem.	141
5.6	L_2 error in the p^* , u^* , and v^* measured at the probe for the moving vortex problem, with no viscosity.	143
5.7	L_2 error in the p^* , u^* , and v^* measured at the probe for the moving vortex problem, with $Re = 100$	143
5.8	L_2 error in the p^* , u^* , and v^* measured at the probe for the moving vortex problem, with $Re = 10000$	143
6.1	Table showing the summary of the error (in brackets) at the resolution 500×500 and order of convergence of various boundary condition methods in the packed-concave domain.	167
6.2	Summary of results for the wave traveling upstream out of the inlet for all the methods. Error at the highest resolution is shown in brackets.	173
6.3	Summary of results for the wave traveling downstream out of the outlet for all the methods. Error at the highest resolution is shown in brackets. .	174
A.1	Different gradient formulations, their dominating error terms, and corresponding order for a uniform arrangement of particle.	195
A.2	Different Laplacian formulations, their dominating error terms, and corresponding order for uniform arrangement of particles.	197
A.3	Different gradient formulations, their dominating quadrature error terms, and corresponding order for uniform arrangement of particles.	199
A.4	Different Laplacian formulations, their dominating quadrature error terms, and corresponding order for uniform arrangement of particles. . . .	200
A.5	Order of quadrature errors for different SPH summations for an irregular arrangement of particles.	202
A.6	Different gradient formulations, their dominating quadrature error terms, and corresponding order for non-uniform particle arrangement.	202

A.7 Different Laplacian formulations, their dominating quadrature error terms, and corresponding order for non-uniform particle arrangement. . . 203

Nomenclature

List of symbols

\mathbf{a}	Acceleration
\mathbf{a}_b	Acceleration due to background pressure
\mathbf{a}_c	Acceleration due to cohesion force
\mathbf{a}_d	Acceleration due to damping force
α	Normalizing factor of a smoothing kernel
\mathbf{a}_{RF}	Acceleration due to repulsion force
A_s	Area of the boundary surface
β	Smoothness of the smoothing kernel near kernel boundary
C	Parametrized curve
c_o	Artificial speed of sound
δ	Damping parameter in δ -SPH scheme
δ_c	Artificial compressibility parameter
$\tilde{\delta}$	Dirac delta distribution
$\partial\Omega$	Boundary of the domain
Δs	Local inter-particle spacing
$\delta\mathbf{u}$	Shifting velocity
Δx	Particle spacing along the coordinate axis
\tilde{d}_i	Magnitude of perturbation

$\tilde{\mathbf{d}}$	Perturbation of the particle position
$\nabla\tilde{W}$	Corrected kernel gradient function
\mathbf{e}	Unit vector
E_i	Error in the approximation
ϵ	Tolerance
F_i	Force on i^{th} particle
F_{\max}	Maximum force on the particles in the domain
F_T	Sum total of the force on the particles in the domain
γ	Scaling factor for cohesion force
h	Support radius of a smoothing kernel
$h_{\Delta x}$	Scaling factor of the kernel
I	Acoustic intensity
I_o	Acoustic intensity threshold
J	Characteristics of the flow
k_r	Repulsion force scaling
l	Number of layers of particles
λ	Curve parameter
m	Mass of the particle
M	Mach number
\mathcal{N}	Set of particles within the support of the kernel
N	Number of particles in the domain
n	Dimension of the space
\hat{n}	Unit normal vector
N_{nbr}	Number of particles in the kernel support

N_s	Number of particles on the boundary surface
\hat{n}_*	Component of unit normal vector
N_t	Number of time steps used in averaging
ν	Kinematic viscosity of the fluid
Ω	Domain on which the field is defined
p	Pressure
p_b	Background pressure
ϕ	Arbitrary particle property
ϕ_{RF}	Repulsion force potential
ψ	Particle density
q	Ratio of the distance from the kernel center and the smoothing length
Re	Reynolds number
ρ	Density of the fluid
σ	Number density
T_r	Relative time taken w.r.t. the minimum time taken for computation
\mathbf{u}	Velocity
U	Maximum velocity in the domain
u	x component of the velocity
u_ϕ	Radial velocity component
$\tilde{\mathbf{u}}$	Transport velocity
v	y component of the velocity
W	Smoothing kernel function
\tilde{W}	Corrected kernel function

\mathbf{x}	Position vector
$\tilde{\mathbf{x}}$	Position vector
ζ	Damping constant

List of acronyms

AC	Artificial Compressibility
ALE	Arbitrary Lagrange Eulerian
BC	Boundary Condition
E-C	Eulerian and Coupled_c viscosity formulation
EDAC	Entropically Damped Artificial Compressibility
EOS	Equation of State
GTVF	Generalized Transport Velocity Formulation
ISPH	Incompressible SPH
L-IPST-C	Lagrangian with Iterative PST and Coupled_c viscosity formulation
L-IPST-F	Lagrangian with Iterative PST and Fatehi_c viscosity formulation
L-IPST-K	Lagrangian with Iterative PST and Korzilius viscosity formulation
L-PST-C	Lagrangian with PST and Coupled_c viscosity formulation
L-RR-C	Lagrangian with Remeshing Regularization and Coupled_c viscosity formulation
LJP	Lennard-Jones Potential
MES	Method of Exact Solutions
MMS	Method of Manufactured Solutions
MOC	Method of Characteristics

MS	Manufactured Solution
NS	Navier-Stokes
NRBC	Non-Reflecting Boundary Condition
PE-IPST-C	P ressure E volution with IPST and Coupled_c viscosity formulation
PLC	Piecewise Linear Curve
PP	Perturbed Periodic
PST	Particle Shifting Technique
RF	Repulsive Force
SOC	Second-Order Convergence/Convergent
SPH	Smoothed Particle Hydrodynamics
TV-C	T ransport V elocity and Coupled_c viscosity formulation
TVF	Transport Velocity Formulation
UP	Unperturbed Periodic
WCSPH	Weakly-Compressible SPH
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

The Smoothed Particle Hydrodynamics (SPH) method was independently developed by Gingold et al. (1977) and Lucy (1977) to simulate astrophysical systems. Since then, it has been used to simulate many fluid and structures problems (Violeau et al., 2016). One of the main advantages of the SPH method is that it can be parallelized easily. As a result, recently it has gained a lot of attention due to rapid development in graphical processing units for general-purpose computing. In the next section, we introduce the SPH method in detail.

1.1 The SPH method

The SPH method can be used for continuous interpolation as well as a discrete approximation. Consider a function f defined in the domain Ω and boundary $\partial\Omega$. The spatial convolution of the function f with the Dirac delta distribution $\tilde{\delta}$ is given by

$$f(\mathbf{x}) = \int_{\Omega} f(\tilde{\mathbf{x}})\tilde{\delta}(\mathbf{x} - \tilde{\mathbf{x}})d^n\tilde{\mathbf{x}}, \quad (1.1)$$

where $\mathbf{x}, \tilde{\mathbf{x}} \in \Omega$ is the domain and $d^n\tilde{\mathbf{x}}$ denotes the infinitesimal volume element, where n is the dimension of the space Ω . The eq. (1.1) reconstructs the field exactly. However, for practical applications, the Dirac delta distribution is replaced by a smoothing kernel W . The modified interpolation reconstructs the function f by using a convolution with a smooth kernel function, given by

$$f(\mathbf{x}) = \int_{\Omega} f(\tilde{\mathbf{x}})W(\mathbf{x} - \tilde{\mathbf{x}}, h)d^n\tilde{\mathbf{x}} + O(h^2), \quad (1.2)$$

where $W(\mathbf{x} - \tilde{\mathbf{x}}, h)$ is a compact kernel function and h is the support radius of the kernel ¹. The kernel has a non-zero value up to the radius kh , where k is a constant parameter. In

¹See derivation in appendix A.4.1.

order to reproduce the given function with $O(h^2)$ accuracy, the kernel function must be symmetric such that

$$W(\mathbf{x} - \tilde{\mathbf{x}}) = W(\tilde{\mathbf{x}} - \mathbf{x}), \quad (1.3)$$

and

$$\int_{\Omega} W(\mathbf{x} - \tilde{\mathbf{x}}, h) d^n \tilde{\mathbf{x}} = 1. \quad (1.4)$$

From the above conditions, we obtain

$$\int_{\Omega} \nabla W(\mathbf{x} - \tilde{\mathbf{x}}, h) d^n \tilde{\mathbf{x}} = \mathbf{0}. \quad (1.5)$$

Additionally, we assume the kernel boundary does not intersect with the domain boundary. The interpolation in eq. (1.2) can be applied to obtain the gradient of the field f , given by

$$\nabla f(\mathbf{x}) = \int_{\partial\Omega} \nabla f(\tilde{\mathbf{x}}) W(\mathbf{x} - \tilde{\mathbf{x}}, h) \mathbf{n}(\tilde{\mathbf{x}}) d\tilde{s} + \int_{\Omega} f(\tilde{\mathbf{x}}) \nabla W(\mathbf{x} - \tilde{\mathbf{x}}, h) d^n \tilde{\mathbf{x}} + O(h^2), \quad (1.6)$$

where $\mathbf{n}(\tilde{\mathbf{x}})$ is the normal at $\tilde{\mathbf{x}}$, $d\tilde{s}$ is the infinitesimal surface at $\tilde{\mathbf{x}}$ on the boundary $\partial\Omega^2$. The first term in eq. (1.6) is equal to zero for a kernel completely inside the domain boundary resulting in

$$\nabla f(\mathbf{x}) = \int_{\Omega} f(\tilde{\mathbf{x}}) \nabla W(\mathbf{x} - \tilde{\mathbf{x}}, h) d^n \tilde{\mathbf{x}} + O(h^2). \quad (1.7)$$

We note that we do not consider the variation due to the change of smoothing radius in the present work. However, the change in smoothing radius can be accommodated by additional terms in the kernel gradient evaluation (Bonet et al., 2005; Muta et al., 2022).

Popular choices for the kernel in the SPH community are

1. Gaussian (Gingold et al., 1977), given by

$$W(q) = \alpha e^{-q^2}, \quad (1.8)$$

where $q = |\mathbf{x} - \tilde{\mathbf{x}}|/h$ and α is a normalizing factor such that $\alpha \int W(q) dq = 1$ is satisfied.

2. Cubic spline (Monaghan et al., 1985), given by

$$W(q) = \alpha \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \leq q \leq 1, \\ \frac{1}{4}(2 - q)^3 & 1 < q \leq 2, \\ 0 & 2 < q. \end{cases} \quad (1.9)$$

²See derivation in appendix A.4.2.

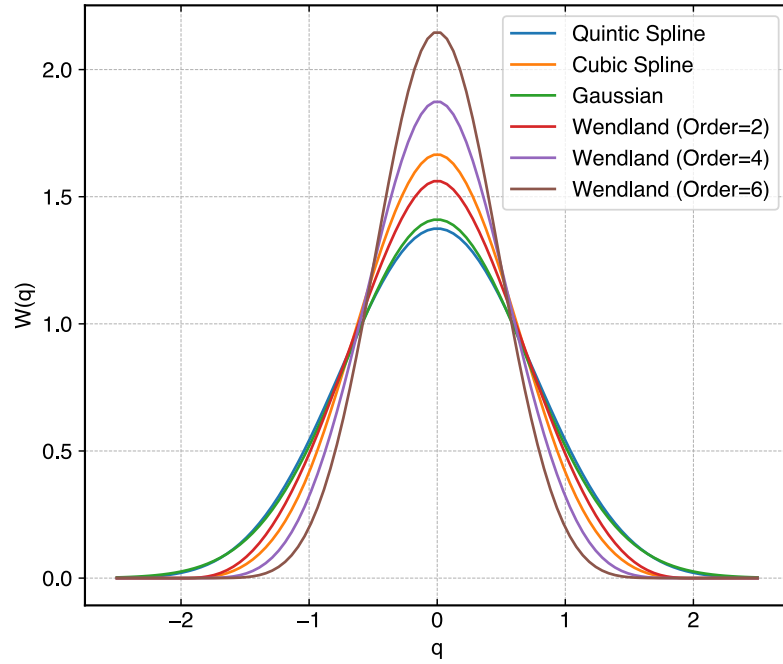


Figure 1.1 : Plots of some of the widely used SPH kernels.

3. Quintic spline (Morris, 1996), given by

$$W(q) = \alpha \begin{cases} (3 - q)^5 - 6(2 - q)^5 + 15(1 - q)^5 & 0 \leq q \leq 1, \\ (3 - q)^5 - 6(2 - q)^5 & 1 < q \leq 2, \\ (3 - q)^5 & 2 < q \leq 3, \\ 0 & 3 < q. \end{cases} \quad (1.10)$$

4. Class of Wendland kernels (Wendland, 1995), where the 4th order kernel is given by

$$W(q) = \alpha \begin{cases} \left(1 - \frac{q}{2}\right)^4 (1 + 2q) & 0 \leq q \leq 2, \\ 0 & 2 < q. \end{cases} \quad (1.11)$$

More details of various other kernels developed in the SPH literature can be found in the book (p. 102) by Liu et al. (2003). In fig. 1.1, we plot all the normalized kernels functions, including Wendland kernels of order 2 and 6 with the change in q .

1.2 Discrete SPH approximation

In the continuous approximation, all of these kernels are $O(h^2)$ accurate (see appendix A.4.1 and appendix A.4.2). In order to approximate an arbitrary field numerically, the domain is discretized into particles with local inter-particle separation Δs , and carrying the required properties like velocity, pressure etc. as shown in fig. 1.2.

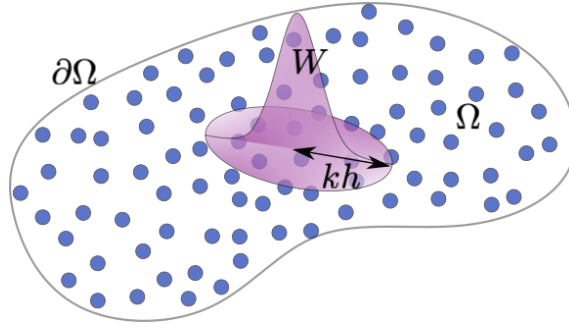


Figure 1.2 : Example of a discretized domain with a kernel.

We define the particle density ψ , given by

$$\psi_i = \sum_{j \in \mathcal{N}(i)} m_j W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (1.12)$$

where m_j is the mass of the j^{th} particle, \mathcal{N} is the set of indices of the particles within the support radius of the kernel. For brevity, we denote $\sum_{j \in \mathcal{N}(i)}$ as \sum_j . We denote the cardinality $n(\mathcal{N})$ of the set \mathcal{N} as N_{nbr} . Therefore, N_{nbr} is the number of neighbor particles of the i^{th} particle. The number density σ of a particle is given by

$$\sigma_i = \sum_j W(\mathbf{x}_i - \mathbf{x}_j, h). \quad (1.13)$$

Therefore, the volume of a particle ω is defined as

$$\omega_i = \frac{1}{\sigma_i} = \frac{m_i}{\psi_i}. \quad (1.14)$$

In SPH, the integral in eq. (1.2) is approximated using a single point quadrature, given by

$$f(\mathbf{x}_i) \approx \sum_j f(\mathbf{x}_j) W(\mathbf{x}_i - \mathbf{x}_j, h) \omega_j. \quad (1.15)$$

Similarly, the gradient of an arbitrary field in eq. (1.7) is approximated as

$$\nabla f(\mathbf{x}_i) \approx \sum_j f(\mathbf{x}_j) \nabla_i W(\mathbf{x}_i - \mathbf{x}_j, h) \omega_j, \quad (1.16)$$

where $\nabla_i W(\mathbf{x}_i - \mathbf{x}_j, h)$ is the gradient of the smoothing kernel w.r.t. \mathbf{x}_i .

On discretizing the domain, in addition to the $O(h^2)$ error, a new error is introduced due to the numerical quadrature (Quinlan et al., 2006), given by

$$\nabla f(\mathbf{x}_i) = \langle \nabla f(\mathbf{x}) \rangle_i + O(h^2) + |f(\mathbf{x}_i)| O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right), \quad (1.17)$$

where $\langle \bullet \rangle_i$ denotes the discrete SPH approximation in eq. (1.16), Δx is the particle spacing along the coordinate axis, and β is the smoothness of the kernel at the edge of its support³.

³See derivation in appendix A.4.5.

The value of $\beta + 1$ is defined as the first non-zero derivative of the kernel at the edge of its support. For example, $\beta = 2$ for cubic spline kernel, and $\beta = 4$ for quintic spline kernel. The ratio $\frac{h}{\Delta x}$ is called the scaling factor denoted by $h_{\Delta x}$. From eq. (1.17), we can see that the value of $h_{\Delta x}$ must be greater or equal to one for accurate approximation. Furthermore, for a particular Δx , the first term suggests decreasing the scaling factor, whereas the second term suggests increasing the scaling factor. Therefore, one needs to find an optimum value of the scaling factor that results in a lower error.

The error terms shown in eq. (1.17) corresponds to a particle distribution on a Cartesian grid with Δx spacing. In the Cartesian arrangement of particles, each particle in the domain has exactly the same configuration of neighboring particles. However, the error may increase when the particles are not uniformly spaced. Consider a small random perturbation $\tilde{\mathbf{d}}_i$ of the i^{th} particle from a Cartesian arrangement, the modified approximation⁴ is given by

$$\nabla f(\mathbf{x}_i) = \langle \nabla f(\mathbf{x}) \rangle_i + O(h^2) + |f(\mathbf{x}_i)| O\left(\frac{\tilde{d}_i}{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right), \quad (1.18)$$

where $\tilde{d}_i = |\tilde{\mathbf{d}}_i|$. We note that the last term has $f(\mathbf{x}_i)$ factor making the approximation unable to reproduce a constant function (Fatehi et al., 2011). In the next section, we will discuss different discretizations used in the SPH literature that have been proposed for improved accuracy of the operators like gradient, divergence, and Laplacian.

1.3 SPH discretization of operators

In general, a partial differential equation involves operators like the gradient, divergence, Laplacian, and other higher-order derivatives. In this work, we focus on incompressible fluid flow problems. These flows are governed by the incompressible Navier-Stokes (NS) equations, given by

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, \\ \frac{d\mathbf{u}}{dt} &= -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}, \end{aligned} \quad (1.19)$$

where \mathbf{u} and p are the velocity and pressure and ν and ρ are the kinematic viscosity and density of the fluid, respectively. The NS equations involve the divergence of velocity, pressure gradient, and Laplacian of velocity. In the next sections, we discuss various SPH discretization proposed to discretize operators present in eq. (1.19).

⁴See derivation in appendix A.4.6.

1.3.1 Divergence discretization

Monaghan (1994) proposed the simplest method to exactly approximate the divergence of a constant velocity field, given by

$$\langle \nabla \cdot \mathbf{u} \rangle_i = \sum_j (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j, \quad (1.20)$$

where $\nabla W_{ij} = \nabla_i W(\mathbf{x}_i - \mathbf{x}_j, h)$. Similarly, some authors (Monaghan, 2005; Sun et al., 2019) use the discretization, given by

$$\langle \nabla \cdot \mathbf{u} \rangle_i = \sum_j (\mathbf{u}_j + \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j. \quad (1.21)$$

Equation (1.21) is also called an asymmetric approximation i.e. $\langle \nabla \mathbf{u} \rangle_{i \rightarrow j} = \langle \nabla \mathbf{u} \rangle_{j \rightarrow i}$ whereas eq. (1.20) is a symmetric formulation i.e. $\langle \nabla \mathbf{u} \rangle_{i \rightarrow j} = -\langle \nabla \mathbf{u} \rangle_{j \rightarrow i}$. Since we consider only the constant mass of each particle, these discretizations do not affect mass conservation. However, the divergence approximation is sometimes required to be the adjoint of the gradient approximation when symplectic (volume preserving) integrators are employed for time integration (see Violeau (2012) for more details).

1.3.2 Gradient discretization

Monaghan et al. (1983) proposed the discretization for the pressure force term, given by

$$\left\langle \frac{\nabla p}{\rho} \right\rangle_i = \sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \nabla W_{ij}. \quad (1.22)$$

The above formulation is symmetric i.e. $\left\langle \frac{\nabla p}{\rho} \right\rangle_{i \rightarrow j} = -\left\langle \frac{\nabla p}{\rho} \right\rangle_{j \rightarrow i}$. Therefore, the force on the particles is equal and opposite. Therefore, eq. (1.22) conserves the linear momentum of the system. Furthermore, Bonet et al. (1999) proved that the formulation in eq. (1.22) is variationally consistent and conserves the angular momentum of the system. However, the force of a constant pressure field does not vanish when eq. (1.22) is employed. Monaghan (1992) proposed the zero-order accurate gradient approximation, given by

$$\langle \nabla p \rangle_i = \sum_j (p_j - p_i) \nabla W_{ij} \omega_j. \quad (1.23)$$

Equation (1.23) can be written in a conservative form, given by

$$\langle \nabla p \rangle_i = \sum_j (p_j + p_i) \nabla W_{ij} \omega_j. \quad (1.24)$$

Equation (1.20) and eq. (1.24) are skew-adjoint ⁵ Similarly, eq. (1.21) and eq. (1.23) are skew-adjoint.

⁵See derivation in appendix A.1.

Fatehi et al. (2011) show that the zero-order discretization in eq. (1.23) has errors of order

$$|\nabla p_i| \left[O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right) + O\left(\frac{\tilde{d}_i}{h}\left(\frac{\Delta x}{h}\right)^{\beta+1}\right) \right], \quad (1.25)$$

where the particles are irregularly arranged⁶. In order to improve the accuracy, Bonet et al. (1999) proposed a kernel-gradient correction such that the gradient of a linear function is captured exactly⁷. The correction matrix is given by

$$B_i = \left(\sum_j \nabla W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \omega_j \right)^{-1}. \quad (1.26)$$

The kernel gradient is multiplied by the correction matrix to obtain a corrected gradient approximation, given by

$$\langle \nabla p \rangle_i = \sum_j (p_j - p_i) B_i \nabla W_{ij} \omega_j. \quad (1.27)$$

The error for the formulation in eq. (1.27) is $|\nabla^2 p_i| O\left(\tilde{d}_i \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$ (see appendix A.4.6). Liu et al. (2006) proposed another correction using the Taylor-series expansion ensuring the consistency of both constant and linear function approximation⁸. The correction matrix is obtained by solving a linear system given by

$$\begin{bmatrix} \sum_l W_{kl} \omega_l & \sum_l x_{lk} W_{kl} \omega_l & \sum_l y_{lk} W_{kl} \omega_l & \sum_l z_{lk} W_{kl} \omega_l \\ \sum_l W_{kl,x} \omega_l & \sum_l x_{lk} W_{kl,x} \omega_l & \sum_l y_{lk} W_{kl,x} \omega_l & \sum_l z_{lk} W_{kl,x} \omega_l \\ \sum_l W_{kl,y} \omega_l & \sum_l x_{lk} W_{kl,y} \omega_l & \sum_l y_{lk} W_{kl,y} \omega_l & \sum_l z_{lk} W_{kl,y} \omega_l \\ \sum_l W_{kl,z} \omega_l & \sum_l x_{lk} W_{kl,z} \omega_l & \sum_l y_{lk} W_{kl,z} \omega_l & \sum_l z_{lk} W_{kl,z} \omega_l \end{bmatrix} \begin{bmatrix} f_k \\ f_{k,x} \\ f_{k,y} \\ f_{k,z} \end{bmatrix} = \begin{bmatrix} \sum_l f_l W_{kl} \omega_l \\ \sum_l f_l W_{kl,x} \omega_l \\ \sum_l f_l W_{kl,y} \omega_l \\ \sum_l f_l W_{kl,z} \omega_l \end{bmatrix}, \quad (1.28)$$

where $W_{kl,\beta} = \frac{\partial W_{kl}}{\partial \beta}$ and $f_{k,\beta} = \frac{\partial f_k}{\partial \beta}$, where $\beta \in \{x, y, z\}$ are the kernel and function partial derivatives in the β direction, $f_k = f(\mathbf{x}_k)$, and $x_{ij} = x_i - x_j$, $y_{ij} = y_i - y_j$, $z_{ij} = z_i - z_j$. On solving eq. (1.28), we obtain a first order consistent function and its gradient. In a similar manner, Rosswog (2015) and Huang et al. (2019) proposed kernel gradient-free corrections for the gradient approximation⁹. They employed the first moment of the kernel instead of the kernel gradient. The correction matrix proposed by Huang et al.

⁶See discussion in appendix A.4.6.

⁷See derivation in appendix A.3.1.

⁸See derivation in appendix A.3.2.

⁹See derivation in appendix A.3.3.

(2019) is given by

$$\begin{bmatrix} \sum_l W_{kl}\omega_l & \sum_l x_{lk}W_{kl}\omega_l & \sum_l y_{lk}W_{kl}\omega_l & \sum_l z_{lk}W_{kl}\omega_l \\ \sum_l W_{kl}\omega_l & \sum_l x_{lk}W_{kl}x_{kl}\omega_l & \sum_l y_{lk}W_{kl}x_{kl}\omega_l & \sum_l z_{lk}W_{kl}x_{kl}\omega_l \\ \sum_l W_{kl}y_{kl}\omega_l & \sum_l x_{lk}W_{kl}y_{kl}\omega_l & \sum_l y_{lk}W_{kl}y_{kl}\omega_l & \sum_l z_{lk}W_{kl}y_{kl}\omega_l \\ \sum_l W_{kl}z_{kl}\omega_l & \sum_l x_{lk}W_{kl}z_{kl}\omega_l & \sum_l y_{lk}W_{kl}z_{kl}\omega_l & \sum_l z_{lk}W_{kl}z_{kl}\omega_l \end{bmatrix} \begin{bmatrix} f_k \\ f_{k,x} \\ f_{k,y} \\ f_{k,z} \end{bmatrix} = \begin{bmatrix} \langle f \rangle_k \\ \langle f \rangle_{k,x} \\ \langle f \rangle_{k,y} \\ \langle f \rangle_{k,z} \end{bmatrix}, \quad (1.29)$$

where $\langle f \rangle_{k,\beta}$ is the β component of the gradient approximation using eq. (1.16). The correction proposed by Rosswog (2015) uses integral approximation resulting in the same correction matrix in eq. (1.29). The application of the kernel gradient correction removes the symmetric nature of the operators, making them non-conservative. Dilts (1999) and recently Frontiere et al. (2017) have suggested that one may correct the function gradient and then symmetrize the corrected kernel gradients to obtain a conservative formulation, given by

$$\left\langle \frac{\nabla p}{\rho} \right\rangle_i = \sum_j m_j \frac{p_j + p_i}{\rho_j \rho_i} (L_i \nabla W_{ij} - L_j \nabla W_{ji}), \quad (1.30)$$

where L_i is the correction proposed by Liu et al. (2006). However, the accuracy improvement shown was marginal. Therefore, a conservative and linear consistent gradient operator is currently not available.

1.3.3 Laplacian discretization

The Laplacian is a challenging operator in the context of SPH. The simplest method to approximate the Laplacian of velocity is the one where the double derivative of the kernel is employed (Monaghan, 2005), given by

$$\langle \nabla^2 \mathbf{u} \rangle_i = \sum_j \mathbf{u}_j \nabla^2 W_{ij} \omega_j. \quad (1.31)$$

However, the double derivatives of the kernel are very sensitive to any particle disorder. Chen et al. (2000) propose an approach by solving a linear system by taking an inner product with each of the double derivatives and taking into account the leading order error terms. Zhang et al. (2004) proposed using the inner product with all the derivatives of the kernel lower and equal to the required derivative and solving the linear system. This generates a system of 10 equations in two dimensions. Korzilius et al. (2017) propose an improvement over the method of Chen et al. (2000) to evaluate the correction term, given by

$$\langle \tilde{\nabla}^2 u \rangle_i = K_i \left(\sum_j (u_j - u_i) \tilde{\nabla}^2 W_{ij} \omega_j - \mathbf{x}_{ij} \cdot \langle \nabla u \rangle_i \tilde{\nabla}^2 W_{ij} \omega_j \right), \quad (1.32)$$

where $\tilde{\nabla}^2 = \left[\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial x \partial y}, \frac{\partial^2}{\partial y^2} \right]^T$ is the operator, $\tilde{\nabla}^2 W_{ij} = \left[\frac{\partial^2 W_{ij}}{\partial x^2}, \frac{\partial^2 W_{ij}}{\partial x \partial y}, \frac{\partial^2 W_{ij}}{\partial y^2} \right]^T$ and K_i is the correction proposed by Korzilius et al. (2017)¹⁰. The gradient $\langle \nabla \mathbf{u} \rangle_i$ in eq. (1.32) is approximated using the linear consistent formulation in eq. (1.27). Schwaiger (2008) also proposed to include high-order terms in the approximation to improve the accuracy near free-surface. Macià et al. (2012) proposed boundary integral SPH formulation and also includes boundary terms that naturally occur in the formulation.

The Laplacian may also be discretized using the first derivative of the kernel using an integral approximation of the Laplacian. This was first suggested by Brookshaw (1985) and has been improved by Morris et al. (1997) and Cleary et al. (1999), given by

$$\langle \nabla^2 \mathbf{u} \rangle_i = \sum_j 2 \frac{(\mathbf{u}_i - \mathbf{u}_j)}{r_{ij}} \mathbf{e}_{ij} \cdot \nabla W_{ij} \omega_j, \quad (1.33)$$

for two-dimensional domain, where $\mathbf{e}_{ij} = \mathbf{e}_i - \mathbf{e}_j$, $\mathbf{e}_{ij} = \mathbf{x}_{ij}/r_{ij}$ and $r_{ij} = |\mathbf{x}_{ij}|$. They employ a finite difference approximation to evaluate the first order derivative and then convolve this with the kernel derivative. This conserves linear momentum as $\langle \nabla^2 \mathbf{u} \rangle_{i \rightarrow j} = -\langle \nabla^2 \mathbf{u} \rangle_{j \rightarrow i}$. However, this approximation does not converge as the resolution increases, especially in the context of irregular particle distributions (see derivation in appendix A.3.4 and section 2.2.1.). Fatehi et al. (2011) propose an improved formulation by accounting for the leading error term, given by

$$\langle \nabla^2 \mathbf{u} \rangle_i = \sum_j 2 \omega_j \left(\frac{(\mathbf{u}_i - \mathbf{u}_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla \mathbf{u} \rangle_i \right) \mathbf{e}_{ij} \cdot \nabla W_{ij}. \quad (1.34)$$

Fatehi et al. (2011) also proposed corrections for this formulation (see appendix A.3.4). These correction makes the method accurate and convergent but makes the approximation non-conservative.

Another method to discretize the Laplacian is the repeated use of a first derivative and this has been used by Bonet et al. (1999) and Nugent et al. (2000), given by

$$\langle \nabla^2 \mathbf{u} \rangle_i = \sum_j (\langle \nabla \mathbf{u} \rangle_j - \langle \nabla \mathbf{u} \rangle_i) \nabla W_{ij} \omega_j. \quad (1.35)$$

This formulation is generally not popular since it shows high frequency numerical oscillations when the initial condition is discontinuous. Recently, Biriukov et al. (2019) show that these oscillations can be removed by employing smoothing near the discontinuity. We derive the error in the formulations discussed in this section in appendix A.4.

In SPH literature, some of these approximations are widely used to discretize the NS equations. In order to numerically solve the incompressible NS equation in eq. (1.19), two

¹⁰See derivation in appendix A.3.5.

methods are widely used viz. the one where weak-compressibility of the fluid is assumed and the second where the fluid is treated as incompressible. In the next section, we discuss both of these methods in detail.

1.4 Incompressible fluid flow methods in SPH

In order to solve the incompressible NS equations, we must impose the incompressibility condition resulting in a divergence-free velocity. However, the divergence-free condition can be relaxed to solve incompressible flow. In this section, we discuss the two widely used incompressible flow physics models used in SPH.

1.4.1 Incompressible SPH

The incompressible SPH (ISPH) model was first introduced by Cummins et al. (1999). In this model, pressure is obtained by the projection method. The velocity field \mathbf{u}^* is decomposed into a curl-free and a divergence-free component, given by

$$\mathbf{u}^* = \mathbf{u} + \Delta t \frac{\nabla p}{\rho_o}, \quad (1.36)$$

where Δt is the time step, \mathbf{u} is the divergence-free component, ρ_o is the fixed fluid density, and ∇p is the curl-free component. The pressure is obtained by taking the divergence of eq. (1.36), given by

$$\nabla \cdot \left(\frac{\nabla p}{\rho_o} \right) = \frac{\nabla \cdot \mathbf{u}^*}{\Delta t}, \quad (1.37)$$

where \mathbf{u}^* is obtained by integrating the momentum equation without considering the pressure gradient, given by

$$\mathbf{u}^* = \mathbf{u}^n + \Delta t \left(\nu \nabla^2 \mathbf{u} + \mathbf{f} \right)^n, \quad (1.38)$$

where \mathbf{f} is the body force, and \mathbf{u}^n is the velocity at the timestep n . On solving eq. (1.37) for pressure p , the intermediate velocity \mathbf{u}^* is corrected using

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \frac{\nabla p}{\rho_o}. \quad (1.39)$$

1.4.2 Weakly-compressible SPH

Chorin (1967) proposed to solve steady state incompressible flow using artificial compressibility. In this method, instead of solving the NS equations given in eq. (1.19), an auxiliary system of equations is solved, given by

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{u}, \quad (1.40a)$$

$$\frac{d\mathbf{u}}{dt} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}, \quad (1.40b)$$

where ρ is a transport property also called artificial density, the pressure is linked to the artificial density using an artificial equation of state (EOS) given by

$$p = \frac{\rho}{\delta_c}, \quad (1.41)$$

where δ_c is an artificial compressibility parameter. Equation (1.40) can be solved numerically, and it produces steady-state solutions when run for a long time. The artificial speed of sound c_o can be evaluated as

$$c_o = \frac{1}{\delta_c^{1/2}}, \quad (1.42)$$

and the Mach number M is given by

$$M = \frac{\max\{|\mathbf{u}_i| \forall i \in \Omega\}}{c_o}. \quad (1.43)$$

It is necessary that $M < 1$. From equation eq. (1.41), eq. (1.42), and eq. (1.43), we have $\rho = M^2 p$ with maximum velocity in the domain equal to $1m/s$.

The governing equations of the weakly-compressible model in eq. (1.40) can be discretized using different formulations discussed in section 1.2. Some of the widely used schemes that use these discretizations are the standard scheme by Morris et al. (1997), δ -SPH by Antuono et al. (2010), Transport Velocity Formulation (TVF) by Adami et al. (2013), Arbitrary Lagrange Eulerian SPH (ALE-SPH) by Oger et al. (2016), Entropically Damped Artificial Compressibility (EDAC) scheme by Ramachandran et al. (2019), δ^+ SPH by Sun et al. (2019), and Eulerian SPH (EWCSFH) by Nasar et al. (2019). We discuss all these schemes in detail in appendix A.5. The accuracy of these schemes is usually shown by comparing with established results. Moreover, the rate of convergence is rarely computed. The computation of the rate of convergence comes under a broader branch of *code verification* (Oberkampf et al., 1995). In the next section, we will discuss different methods used to verify convergence and the currently used verification techniques in SPH.

1.5 Verification of an implementation

Convergence study of an implementation of a numerical method is of utmost importance. In a convergence study, the resolution of the domain discretization is increased to obtain more accurate solutions. The rate of decrease of error with the resolution is called the rate of convergence of the solution. Convergence validates that the numerical solution tends to be the actual solution of the governing differential equation.

Oberkampf et al. (1995) formally introduced the notion of verification and validation for computational codes. The definitions of validation and verification by Roache (1998) are widely accepted. Verification is a mathematical exercise wherein we assess if

the implementation of a numerical method is consistent with the chosen governing equations. For example, verification will allow us to check whether the code of a second-order accurate method is indeed second-order. On the other hand, validation tests whether the chosen governing equations suitably model the given physics. This is often established by comparison with the results of experiments.

According to Roy (2005), verification can be further classified into two categories: code verification and solution verification. The code is tested for its correctness in code verification, whereas in solution verification, we quantify the errors in the solution obtained from a simulation. For example, in solution verification, we solve a specific problem and estimate the error through some means like a grid convergence study. In the next section, we present a literature survey of various theoretical and numerical attempts to obtain the accuracy or the rate of convergence of the SPH operators and schemes.

1.5.1 Literature survey

Continuous SPH interpolation involves errors that are $O(h^2)$. However, for discrete SPH approximation, second-order convergence is a grand challenge (Vacondio et al., 2020). Various attempts have been made to obtain a second-order rate of convergence by manipulating the smoothings (or support) radius h . Hernquist et al. (1989) proposed that the support radius h be increased such that $h \propto \Delta x^{-1/3}$ in three dimensions. Subsequently, Quinlan et al. (2006) derived error estimates for the standard SPH discretization and found that the ratio $h/\Delta x$ must increase as the h value is reduced to attain convergence; this is due to the $O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$ term in eq. (1.17). This term is an issue because as h increases, the number of neighbors for each particle increases resulting in a prohibitive increase in computational effort. Furthermore, increasing the smoothing radius also reduces the accuracy of the method. This is the approach used in the work of Zhu et al. (2015), who proposed that the number of neighbors $N_{nbr} \propto N^{0.5}$ to get convergence using SPH kernels, where N is the number of particles in the domain. However, when we decrease $h_{\Delta x}$, the accuracy of the approximation improves without changing the resolution. Furthermore, one cannot have the scaling factor $h_{\Delta x} < 1$ as the number of particles in support of a given particle decreases.

Many authors investigated the effect of the smoothness of the particle distribution and kernel function. Kiara et al. (2013b) shows that when the particles are distributed “uniformly” such that the perturbation of particles from a Cartesian arrangement is small, it is possible to obtain second-order convergence. The results of Quinlan et al. (2006) show that when using sufficiently smooth kernels (where β is large or infinite), one can obtain second-order convergence for particle distribution on a Cartesian grid. Indeed,

Lind et al. (2016) demonstrate that one can obtain higher-order convergence using higher-order kernels for particle distributions on a Cartesian mesh. Evidently, the second-order convergent approximation can be achieved in special cases.

However, for kernels normally used in SPH, the SPH approximations of derivatives become inaccurate even on a uniform grid unless a very large smoothing radius is used. In section 1.3.2 and section 1.3.3, we discussed many methods that have been proposed to correct the discrete operators. These typically ensure that the derivative approximation of a linear function is exact. These corrections make the derivative approximation second-order accurate but increases the computational cost of the gradient computation two-fold. Quinlan et al. (2006) showed second-order convergence of a sinusoidal function and its derivative approximation using standard SPH method with the change in $h_{\Delta x}$. Schwaiger (2008) showed improved accuracy by approximating the Laplacian of various 2D functions. Macià et al. (2012) demonstrated the accuracy by showing the maximum absolute error in constant, linear, and square functions with the change in the number of particles in 2D. However, in SPH literature, a complete comparison of different formulations has yet to be performed.

As mentioned earlier, the accuracy of different schemes proposed in SPH literature is shown using a comparison test. They show convergence in the form of plots that approach an exact or experimental solution with increasing resolution without formally computing the order of convergence. Chen et al. (2000) demonstrated the accuracy by solving Burgers' equations in one and two dimensions. Zhang et al. (2004) show convergence by solving one-dimensional wave equation and two-dimensional heat conduction problem. Marrone et al. (2011) solved problems like dam-break and compared the accuracy using the experimental data. Kiara et al. (2013a) demonstrate the correctness of the implementation by solving a hydrostatic problem, the collapse of a liquid column and dam-break problem. Adami et al. (2013) show the convergence plot by comparing the results with the exact solution of the Taylor-Green problem. Huang et al. (2019) also solved the Taylor-Green problem to demonstrate the accuracy of their method. Sun et al. (2019) compared the decay of the kinetic energy of the system with the change in resolution for the Taylor-Green problem. However, they need to formally compute the rate of convergence. Therefore, a rate of convergence study is rarely performed in SPH.

Some authors do perform a limited convergence study. Dehnen et al. (2012) show the rate of convergence by solving the Gresho-Chan vortex problem; however, the highest order of convergence achieved was 0.6. Rosswog (2015) also solved the Gresho-Chan vortex problem and achieved a rate of convergence 1.0. Frontiere et al. (2017) achieve 1.9 rate of convergence for the 1D acoustic wave problem. Ramachandran et al. (2019) show

a first-order convergence that gradually reduces at a higher resolution for the Taylor-Green problem.

Some authors demonstrate second-order convergence for simpler problems with particles uniformly placed on a Cartesian grid configuration. Schwaiger (2008), Cleary et al. (1999), Fatehi et al. (2011), and Korzilius et al. (2017) demonstrated convergence by solving the heat conduction problem, whereas Macià et al. (2012) solved the Poisson equation. Lind et al. (2016) and Nasar et al. (2019) showed more than second-order convergence; however, an Eulerian formulation is employed.

To the best of our knowledge, none of the contemporary Lagrangian SPH schemes appear to demonstrate a formal second-order convergence even for simple fluid mechanics problems like the Taylor-Green vortex problem for which an exact solution is known. This motivates the current work on the development of a second-order convergent WCSPH scheme.

1.6 Objective of current work

In this work, we carefully construct a family of Lagrangian WCSPH schemes that demonstrate second-order numerical convergence in a periodic domain. Subsequently, we identify convergent boundary condition implementations to develop a code that can simulate a wind tunnel-like simulation with high accuracy using the weakly-compressible model. The following are the main objectives of the work.

1. Compare the convergence of state-of-the-art WCSPH schemes by solving a periodic problem with an exact solution.
2. Identify the issues that prevent the convergence of the WCSPH schemes present in the literature and develop an accurate second-order scheme.
3. Develop a fast and robust technique to test the convergence of a WCSPH code using the Method of Manufactured Solutions (MMS).
4. Develop parameter-free, accurate, non-reflecting inlet and outlet boundary condition implementations.
5. Identify convergent boundary condition implementations for solid, inlet, and outlet boundaries.
6. Develop an algorithm to simulate a flow past a circular cylinder with a high level of accuracy.

In the interest of reproducibility, we implement all the schemes using the PySPH (Ramachandran et al., 2021) framework, and all the results shown in this work is automatically generated through the use of an automation framework `automan` (Ramachandran, 2018). The source code for the work is available at various repositories available at <https://gitlab.com/pypr>.

1.7 Outline

In chapter 2, we show the comparison of various aspects involved in an SPH scheme starting from the kernel to various discretizations for gradient, divergence, and Laplacian approximation. We then discuss the new second-order convergent scheme and its variations. We verify the convergence and conservation properties by solving different problems using new and existing schemes. In chapter 3, we discuss the use of the MMS to verify WCSPH schemes. We show how one can use the MMS within the framework of Lagrangian formulations. In chapter 4, we discuss various methods to capture the features of a solid body using multiple layers of particles for an accurate boundary condition implementation. In chapter 5, we discuss various existing boundary condition implementations for Neumann pressure, slip, no-slip, and inflow/outflow boundaries. We propose a hybrid non-reflecting outlet boundary condition implementation and novel test cases testing various aspects of the implementation. In chapter 6, we propose various manufactured solutions to verify the convergence of various boundary condition implementations. We use the proposed convergent schemes and the boundary condition implementations to propose a complete second-order convergent scheme. In chapter 7, we discuss our contributions and the future scope of the work.

Chapter 2

Construction of second-order convergent WCSPH schemes

As discussed in the previous chapter, the convergence of the WCSPH schemes is a grand challenge (Vacondio et al., 2020). Many authors have proposed schemes like TVF, EDAC, δ -SPH, δ^+ SPH, and EWCSH (see appendix A.5 for details). They show the accuracy of the obtained solutions by comparing them with analytical or established results. However, the rate of convergence of these schemes with the increase in particle resolution is not investigated. In this chapter, we systematically investigate various aspects of a weakly-compressible SPH scheme viz. type of kernel function, the smoothing scale $h_{\Delta x}$, application of kernel gradient correction, and the type of discretization formulation.

We first study several commonly used SPH kernels in the context of function and derivative approximation using particles either in a Cartesian arrangement or in an irregular but packed configuration of particles. The packed configuration is created by shifting the particles from an irregular distribution such that the particle density is uniform. We choose a suitable kernel gradient correction scheme that produces second-order convergence for a function gradient approximation. We then select a suitable smoothing kernel and smoothing radius based on this study. We then systematically study the various discretization operators along with suitable corrections. Our investigations are in two dimensions, although the results are also applicable in three dimensions. Our numerical investigation covers a wide range of resolutions, with our highest resolution having a quarter million particles with $\frac{L}{\Delta x} = 500$, where $L = 1m$ is the length of the domain.

Once we have identified suitable second-order convergent operators, we carefully construct SPH schemes that display a second-order convergence (SOC). We use the Method of Exact Solutions (MES), and solve the Taylor-Green vortex problem to demonstrate this. We also compare our results with those of several established SPH methods

currently used. We study the accuracy and convergence and investigate the computational effort required. We construct both Lagrangian and Eulerian schemes that are fully second-order convergent. We provide schemes that use either an artificial compressibility in the form of an equation of state as done in the standard WCSPH scheme or using a pressure evolution equation similar to the EDAC scheme. Once we have demonstrated second-order convergence for the Taylor-Green vortex problem, we proceed to investigate the schemes using the Gresho-Chan vortex (Gresho et al., 1990) problem as well as an incompressible shear layer problem (Di et al., 2005) and look at how the lack of manifest conservation impacts the conservation of linear and angular momentum.

2.1 Selection of approximating kernel

In this section, we compare various SPH kernels for their accuracy and order of convergence in a discrete domain. We evaluate the error in function and derivative approximation in a two-dimensional periodic domain. We simulate periodicity by copying the appropriate particles and their properties near the boundary such that the boundary particles have full support (Randles et al., 1996). The particles are either placed in a uniform Cartesian mesh (unperturbed) or in a *packed* arrangement referred to as unperturbed periodic (UP) or perturbed periodic (PP), respectively. In order to obtain the packed configuration, the particles are slightly perturbed from a uniform mesh, and their positions are moved and allowed to settle into a particle distribution with a nearly constant density using a particle packing algorithm (Colagrossi et al., 2012). The algorithm effectively ensures that the particles are not clustered and have minimal number density variations. This distribution can be achieved using any Particle Shifting Technique (PST) (Colagrossi et al., 2012; Huang et al., 2019; Lind et al., 2012). In fig. 2.1, we show both domains.

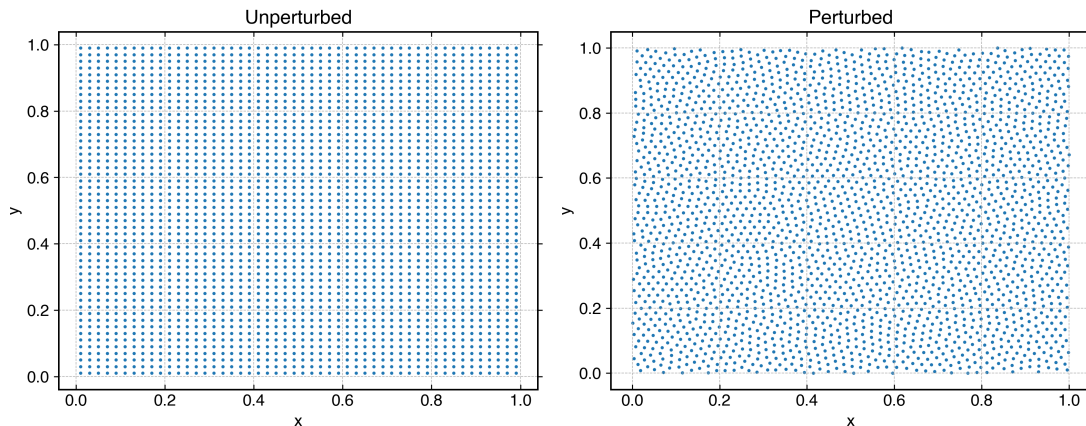


Figure 2.1 : The unperturbed periodic particle and perturbed periodic particles.

Name	Radius	β	Remark
G : Gaussian	3	0	Truncated for low N_{nbr}
QS : Quintic spline	3	2	Tensile instability
CS : Cubic spline	2	4	Pairing and tensile instability
WQ_2 : Wendland $O(2)$	2	3	No tensile or pairing instability
WQ_4 : Wendland $O(4)$	2	5	Produces higher accuracy
WQ_6 : Wendland $O(6)$	2	7	Produces higher accuracy

Table 2.1 : Kernels and their properties (β reported for 1D kernels)

We consider the set of kernels discussed in chapter 1 also listed in table 2.1. It covers a wide range, including high order, kernels having tensile instability and pairing instability (see appendix A.2 for details). In order to assess the effect of $h_{\Delta x}$ for a kernel, we perform the numerical experiment proposed by Dehnen et al. (2012). We evaluate particle density ψ using eq. (1.12) for increasing number of neighbors N_{nbr} , for each of the kernels. The increase in N_{nbr} corresponds to the scaling of the smoothing kernel using the $h_{\Delta x}$ parameter. In this numerical experiment, we change both the resolution and $h_{\Delta x}$ to increase N_{nbr} . Since we consider a uniform particle distribution, we compute the absolute error in the density for only one particle, given by

$$L_1 = |\psi_i - 1.0|, \quad (2.1)$$

where i is any particle having full kernel support.

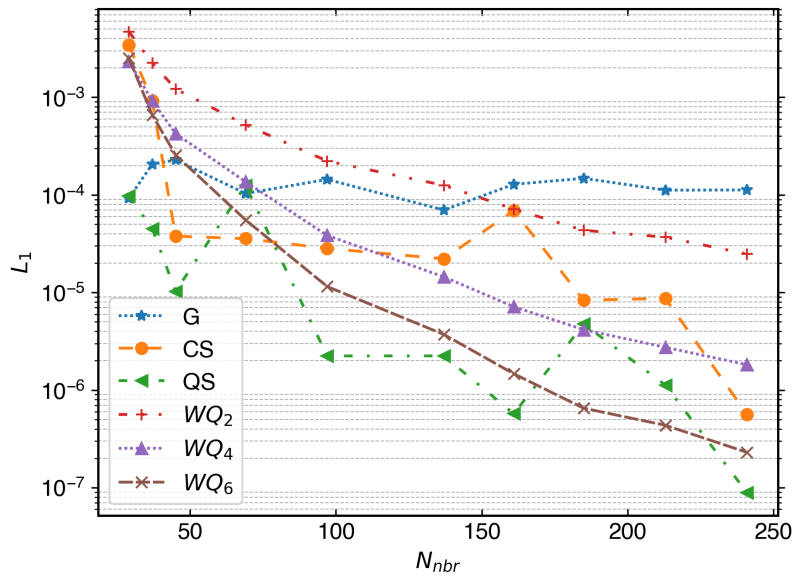


Figure 2.2 : The particle density for different kernels with varying numbers of neighbors.

In fig. 2.2, we plot the absolute error in the particle density of particles in a UP domain for different kernels with the change in N_{nbr} under the kernel support. The Wendland kernels show a monotonic decrease in error with the increasing N_{nbr} . However, in the case of the G and QS kernels, the errors are an order less at a lower N_{nbr} compared to the Wendland class of kernels. The error in the G kernel does not change significantly with the change in the N_{nbr} compared to others. It is because we truncate the G kernel to have compact support. In the case of the QS kernel, the error is lower than the WQ_4 kernel in the plot. Therefore, we drop WQ_2 and WQ_4 kernels in the subsequent investigations since it reaches the order of accuracy of QS when N_{nbr} is approximately 60. High N_{nbr} results in higher computational costs.

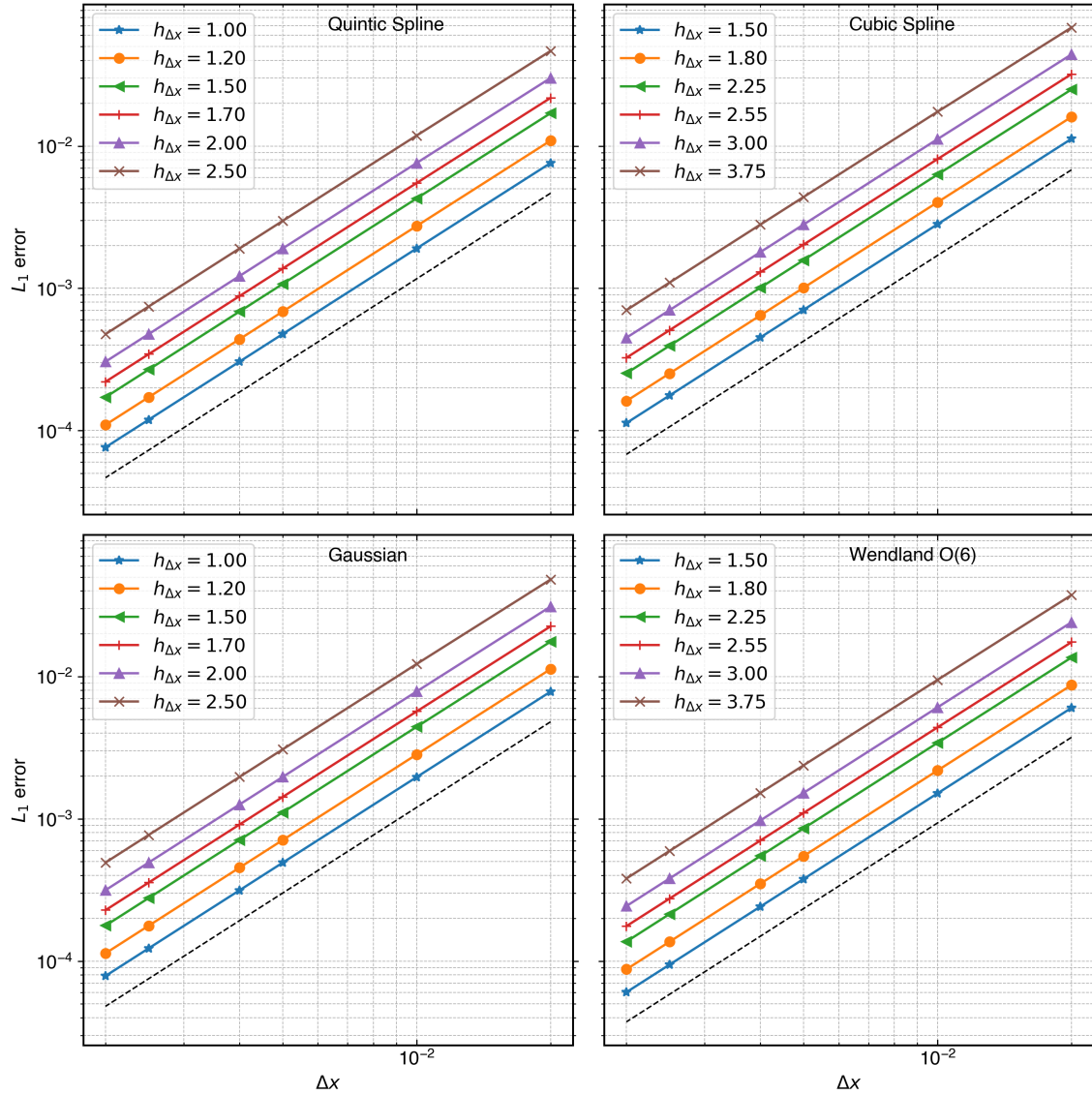


Figure 2.3 : Order of convergence for the approximation of a function for different $h_{\Delta x}$ values in a UP domain. The dashed line shows the second-order rate.

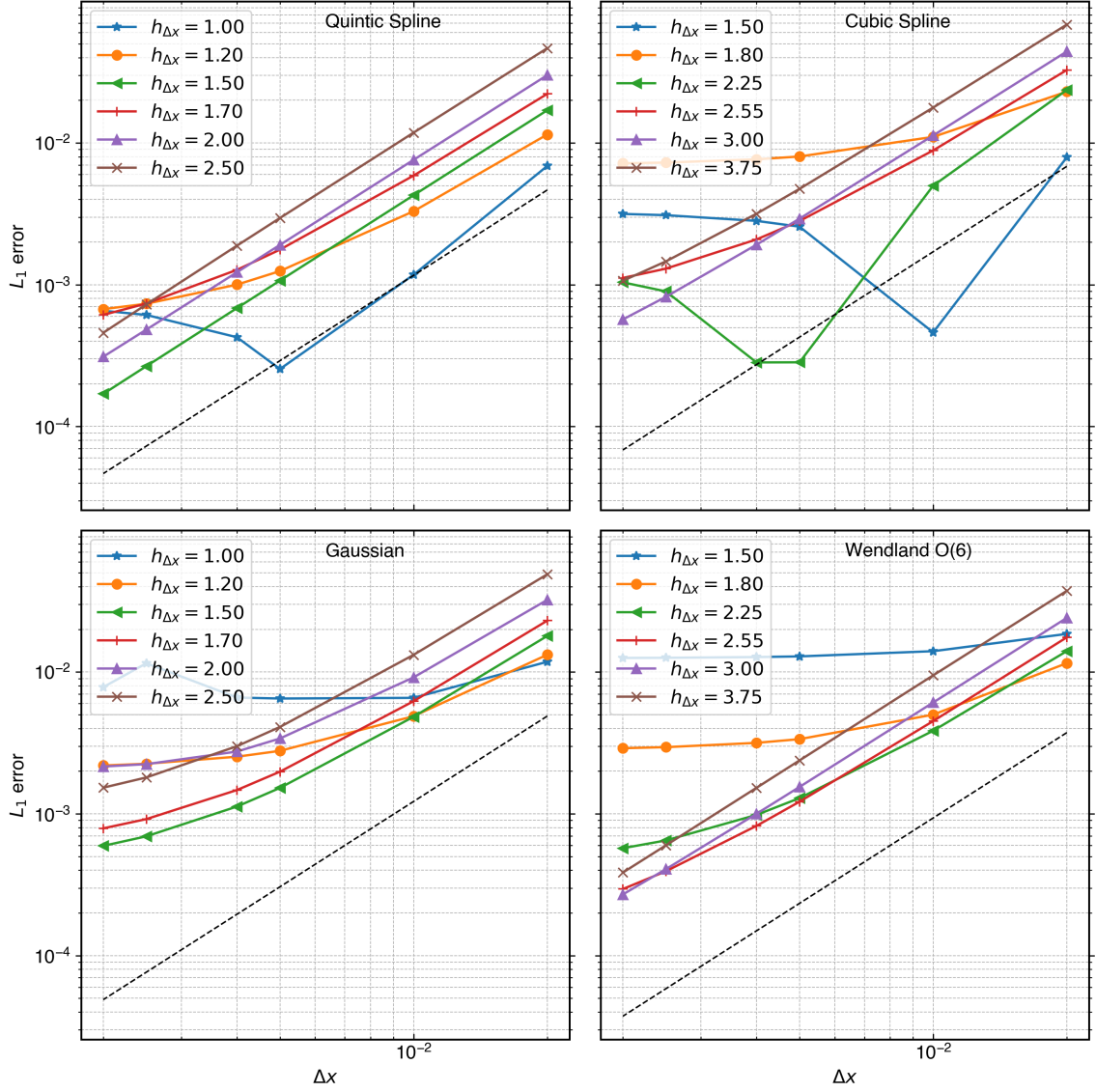


Figure 2.4 : Order of convergence for derivative approximation for different $h_{\Delta x}$ values in a UP domain. The dashed line shows the second-order rate.

We compare the four kernels viz. G , CS , QS , and WQ_6 for the convergence of the function and its gradient approximation. We consider the field

$$f = \sin(\pi(x + y)). \quad (2.2)$$

Given a function g_o and its approximation g , we evaluate the L_1 error using

$$L_1 = \frac{\sum_i^N |g(\mathbf{x}_i) - g_o(\mathbf{x}_i)|}{\sum_i^N |g_o(\mathbf{x}_i)|}, \quad (2.3)$$

where N is the total number of particles in the domain. Since the CS and WQ_6 kernels have a support radius of $2h$ whereas, the G and QS kernels have support radius of $3h$, we set the $h_{\Delta x}$ such that the N_{nbr} is same in the UP domain. Therefore, when $h_{\Delta x} = 1.0$ for

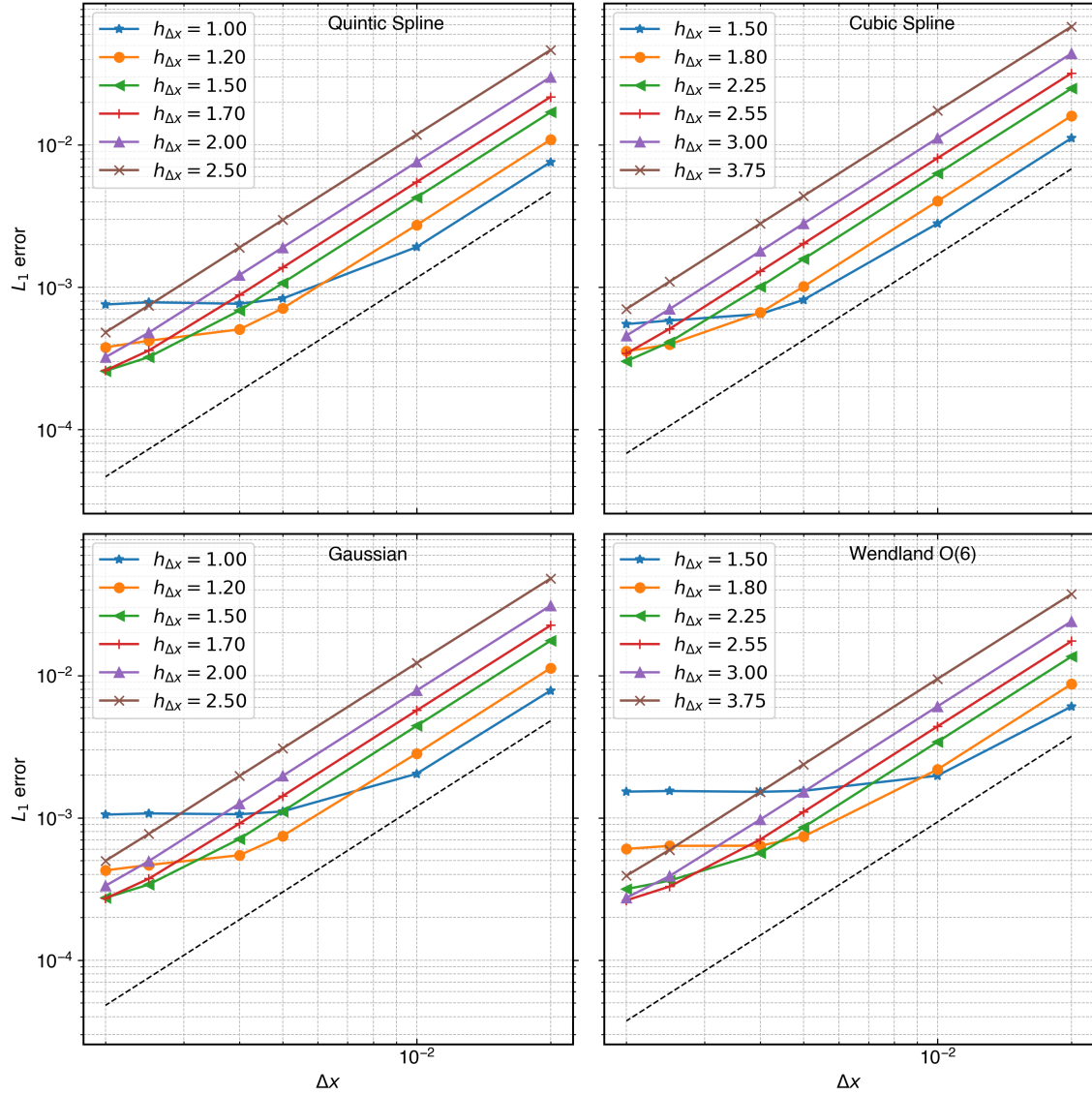


Figure 2.5 : Order of convergence for function approximation for different $h_{\Delta x}$ values in a PP domain. The dashed line shows the second-order rate.

QS (or G) kernel, we take $h_{\Delta x} = 1.5$ for the CS (or WQ_6) kernel. For the convergence study, in this work, we consider 50×50 , 100×100 , 200×200 , 250×250 , 400×400 , and 500×500 resolutions for all the test cases unless stated otherwise.

2.1.1 Unperturbed periodic domain

In fig. 2.3, we plot the L_1 error in the function approximation as a function of the resolution for different values of $h_{\Delta x}$ in a UP domain. We observe similar error values for all the kernels except CS . We obtain second-order convergence (SOC) in a UP domain up to a considerably high resolution of 500×500 , as expected for all the kernels.

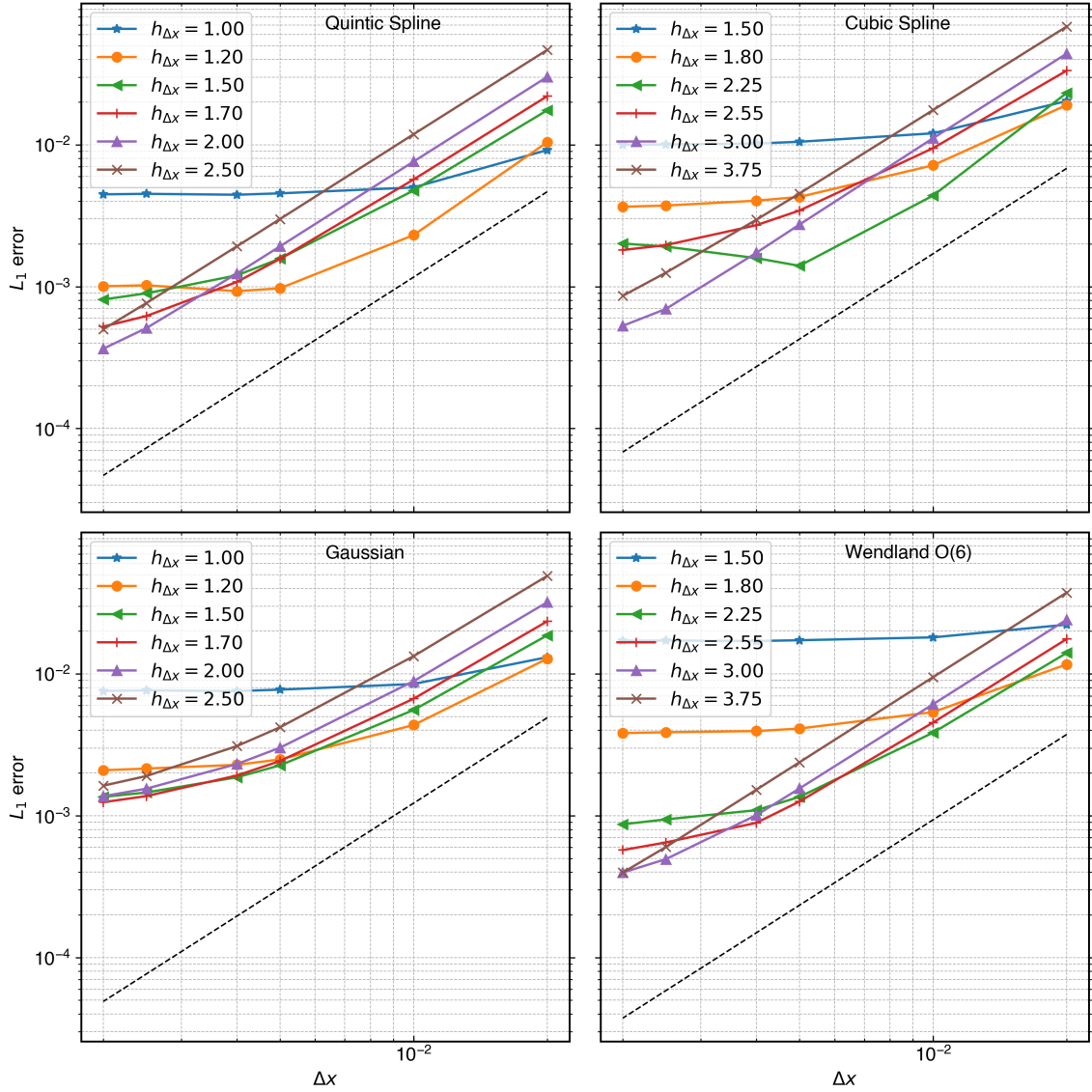


Figure 2.6 : Order of convergence for the derivative approximation for different $h_{\Delta x}$ values in a PP domain. The dashed line shows the second-order rate. Equation (1.23) is used for the approximation.

In fig. 2.4, we plot the L_1 error in the derivative approximation of the function in eq. (2.2) in a UP domain. The G and QS kernels show a better convergence rate compared to CS and WQ_6 for lower $h_{\Delta x}$. The G kernel does not show SOC even at $h_{\Delta x} = 2.5$ since we use a truncated Gaussian. The CS and WQ_6 kernel show SOC only when $h_{\Delta x} \geq 3.0$. The QS kernel shows SOC at $h_{\Delta x} > 1.5$ however, a reasonable convergence can be seen for $h_{\Delta x} = 1.2$ as well.

2.1.2 Perturbed periodic domain

In an SPH simulation, the particles advect with different velocities, and thus the distribution of particles does not remain like a UP domain. However, any particle shifting technique can make the particle distribution uniform. Thus, it is essential to observe the convergence rate in the PP domain as well.

In fig. 2.5, we plot the L_1 error of the approximation of the field given in eq. (2.2) as a function of resolution in a PP domain for different $h_{\Delta x}$ values. The convergence rates tend to zero for higher resolution for a low value of $h_{\Delta x}$ for all the kernels. The WQ_6 kernel performs worse than the CS kernel at lower $h_{\Delta x}$ values; however, the errors are significantly lower in WQ_6 when the $h_{\Delta x}$ value increase. On comparing G and QS , the error plot looks exactly the same except when $h_{\Delta x} = 1.0$.

The SPH approximation of the gradient of a function is not even zero-order accurate in a perturbed domain (see derivation in appendix A.4.3). The derivatives diverge when we evaluate it using eq. (1.17). We use a zero-order consistent formulation in eq. (1.23) to compare the kernels.

In fig. 2.6, we plot the L_1 error in the function derivative approximation as a function of resolution using eq. (1.23) in a PP domain for different $h_{\Delta x}$ values and kernels. Clearly, the approximation for all the kernels shows poor convergence. The G kernel does not show SOC for high $h_{\Delta x}$, which is the same as observed in the case of the UP domain. The accuracy in the case of QS and CS oscillates when going from lower $h_{\Delta x}$ to higher values. Zhu et al. (2015) suggest increasing the $h_{\Delta x}$ as one increases the resolution, but given the inconsistent behavior of the CS and QS kernels, these may not be suitable for these kernels. The zero-order convergence rate occurs due to the dominance of discretization error (the term $\left(\frac{\Delta x}{h}\right)^{\beta+1}$ in eq. (1.17)) when the resolution increases in the PP domain.

In order to study the effect of kernel gradient correction, we apply the correction proposed by Bonet et al. (1999) for all the selected kernels (see appendix A.3.1). The function gradient is computed using the corrected formulation in eq. (1.27). In fig. 2.7, we plot the L_1 error in the derivative approximation as a function of resolution. Clearly, all the kernels Gaussian (G), Wendland O(6) (WQ_6), cubic spline (CS), and quintic spline (QS) show more or less the same behavior. Thus, we can choose any of these kernels for our convergence study of the WCSPH schemes. In fig. 2.7, it can be seen that the WQ_6 and G kernels do not sustain second-order behavior. Furthermore, a lower $h_{\Delta x}$ results in higher quadrature error; whereas, a higher $h_{\Delta x}$ increases the number of particles in the support radius resulting in higher computational cost. Therefore in this work, we heuristically choose the QS kernel with a $h_{\Delta x} = 1.2$ for all the test cases henceforth.

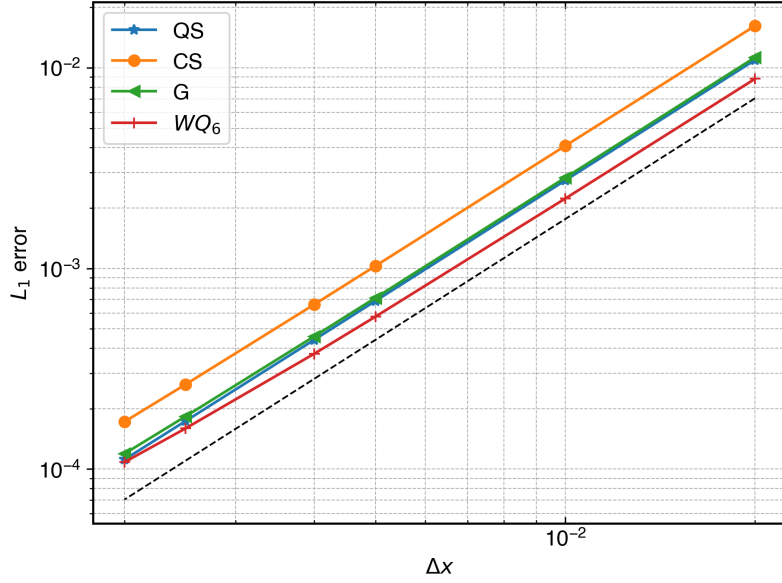


Figure 2.7 : The convergence of the derivative approximation with different kernels when the kernel gradient correction of Bonet et al. (1999) is used on a PP domain. We use $h_{\Delta x} = 1.2$.

2.2 Comparison of discretization operators

There are many different ways to discretize the operators present in the NS equations, as discussed in chapter 1. In this section, we compare the rate of convergence of various formulations discussed in section 1.3. One of the key features of these formulations was pair-wise symmetry and asymmetric. The pair-wise asymmetry results in linear momentum conservative, whereas the divergence and gradient operators being adjoint result in volume-preserving time integration. In view of this, we consider all formulations, irrespective of their unique property, in the present study. In this section, we show the convergence with h instead of Δx since we have fixed the value of $h_{\Delta x} = 1.2$ unless stated otherwise.

2.2.1 Comparison of $\nabla \cdot \mathbf{u}$ approximation

We rewrite the zero-order consistent SPH approximation for the divergence operator as described in section 1.3.1, given by

$$\langle \nabla \cdot \mathbf{u} \rangle_i = \sum_j (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j. \quad (2.4)$$

We refer to the approximation given in eq. (2.4) as div . We apply the kernel gradient correction as done in eq. (1.27) for a first-order consistent approximation, given by

$$\langle \nabla \cdot \mathbf{u} \rangle_i = \sum_j (\mathbf{u}_j - \mathbf{u}_i) \cdot B_i \nabla W_{ij} \omega_j. \quad (2.5)$$

We refer to the corrected form as `div_bc`.

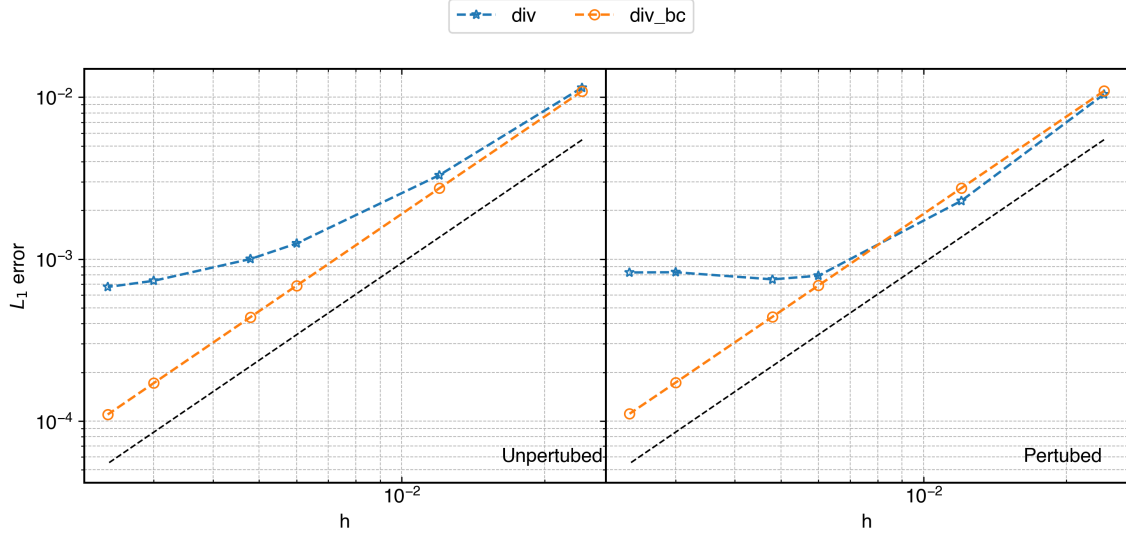


Figure 2.8 : The rate of convergence in UP (left) and PP (right) domains for velocity divergence approximation in eq. (2.4). The dashed line shows the SOC rate. The suffix `_bc` represents the corresponding form with Bonet correction.

We consider the velocity field $\mathbf{u} = \sin(\pi(x+y))(\hat{\mathbf{i}} + \hat{\mathbf{j}})$. The divergence of the velocity $\nabla \cdot \mathbf{u} = 2\pi \cos(\pi(x+y))$. We evaluate the L_1 error in the approximation using eq. (2.3). In fig. 2.8, we plot the L_1 error in the divergence approximation in a UP and PP domain. The uncorrected approximation does not display SOC since the discretization error dominates as we approach higher resolutions. Clearly, the corrected form shows SOC even in the case of a PP domain.

In order to evaluate the accuracy of the approximation in a divergence-free field, we consider the velocity field

$$\begin{aligned} u &= -\cos(2\pi x) \sin(2\pi y), \\ v &= \sin(2\pi x) \cos(2\pi y). \end{aligned} \quad (2.6)$$

In fig. 2.9, we plot the L_1 error using eq. (2.3) in the divergence computation as a function of resolution for the UP and PP domains. Clearly, the divergence is zero in a UP domain owing to the symmetry of the particles. However, the error in the PP domain remains about the same order as seen in the case of the general field in fig. 2.8. We discuss this dependency of the approximation on the velocity field and the particle arrangement in section 2.3.2. However, we note that the Bonet correction does not correct this issue. We observe the implication of this behavior when we compare the schemes in section 2.4.3.

The continuity equation corresponds to the mass conservation of the system, since the mass of each particle is kept constant; we implicitly satisfy the global conservation of mass.

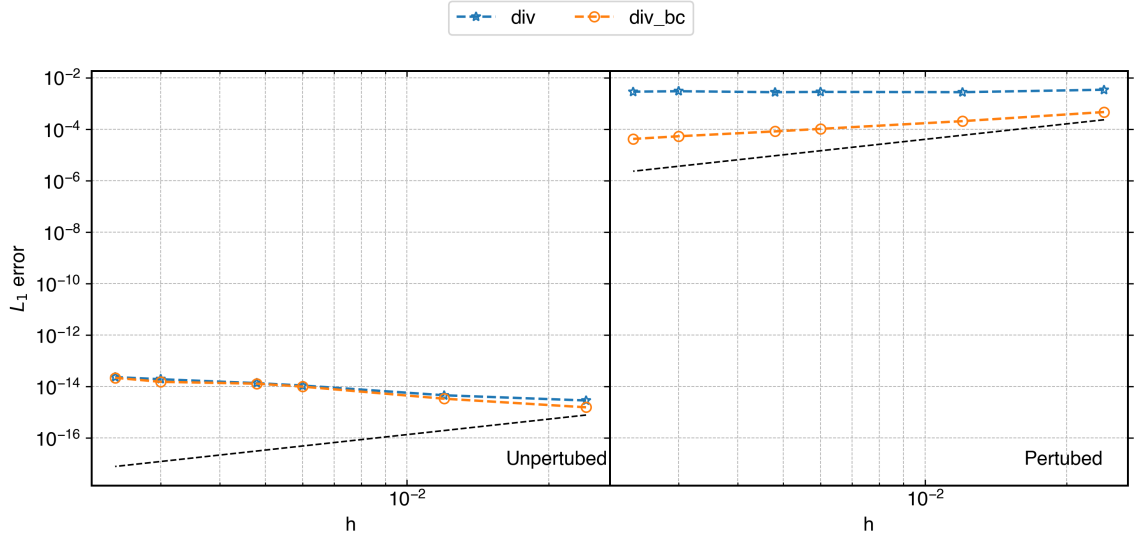


Figure 2.9 : The rate of convergence in UP (left) and PP (right) domains for divergence approximation of a divergence-free velocity field. The dashed line shows the SOC rate. The suffix `_bc` represents the corresponding form with Bonet correction.

2.2.2 Comparison of $\frac{\nabla p}{\rho}$ approximation

In this section, we compare various pressure gradient approximations. In table 2.2, we list the gradient approximations considered in this study. The `sym1` and `sym2` are the symmetric, conservative form of the gradient approximation. The `asym` is the asymmetric form. Since the SPH gradient approximation does not show SOC in a perturbed domain (See appendix A.4.6), we also consider the kernel gradient correction employed for each of the approximations. In this work, we refer to the correction proposed by Bonet et al. (1999) as *Bonet correction* and the one proposed by Liu et al. (2006) as *Liu correction*. We add the suffix `_bc`, and `_lc` respectively in the plots and tables to indicate these corrections. The application of corrections renders the symmetric forms non-conservative. One can employ the method of symmetrization of the kernel proposed by Dilts (1999) to make it conservative again. We refer to this formulation as `sym_sl`.

In order to compare the convergence, we consider a pressure field $p = \sin(\pi(x + y))$. We determine the L_1 error using eq. (2.3), where $g(\mathbf{x}_i)$ is the pressure gradient evaluated using the approximation and $g_o(\mathbf{x}_i)$ is the exact pressure gradient. The exact pressure gradient $\nabla p = \pi \cos(\pi(x + y))(\hat{\mathbf{i}} + \hat{\mathbf{j}})$. We compare only the x-component of the results. In fig. 2.10, we plot the error in the various gradient approximations discussed above in both UP and PP domains. In the UP domain, barring the `sym2_lc`, all the corrected gradient approximations behave the same, whereas the uncorrected gradients do not display SOC. The corrected versions retain SOC even at high resolution since it reduces the dis-

Name	Expression	Used in
asym_bc	$\sum_j \frac{(p_j - p_i)}{\rho_i} B_i \nabla W_{ij} \omega_j$	Hashemi et al. (2012)
asym	$\sum_j \frac{(p_j - p_i)}{\rho_i} \nabla W_{ij} \omega_j$	Sun et al. (2018)
sym1_bc	$\sum_j \frac{(p_j + p_i)}{\rho_i \rho_j} B_i \nabla W_{ij} m_j$	-
sym1_lc	$\sum_j \frac{(p_j + p_i)}{\rho_i \rho_j} L_i \nabla W_{ij} m_j$	-
sym1	$\sum_j \frac{(p_j + p_i)}{\rho_i \rho_j} \nabla W_{ij} m_j$	δ^+ SPH
sym2_bc	$\sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) B_i \nabla W_{ij}$	-
sym2_lc	$\sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) L_i \nabla W_{ij}$	-
sym2	$\sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \nabla W_{ij}$	WCSPH, TVF, EDAC, ISPH
sym_sl	$\sum_j m_j \frac{p_j + p_i}{\rho_j \rho_i} (L_i \nabla W_{ij} - L_j \nabla W_{ji})$	Frontiere et al. (2017)

Table 2.2 : Different gradient approximations for $\frac{\nabla p}{\rho}$. The formulations with blank “Used in” columns are not used in any existing scheme.

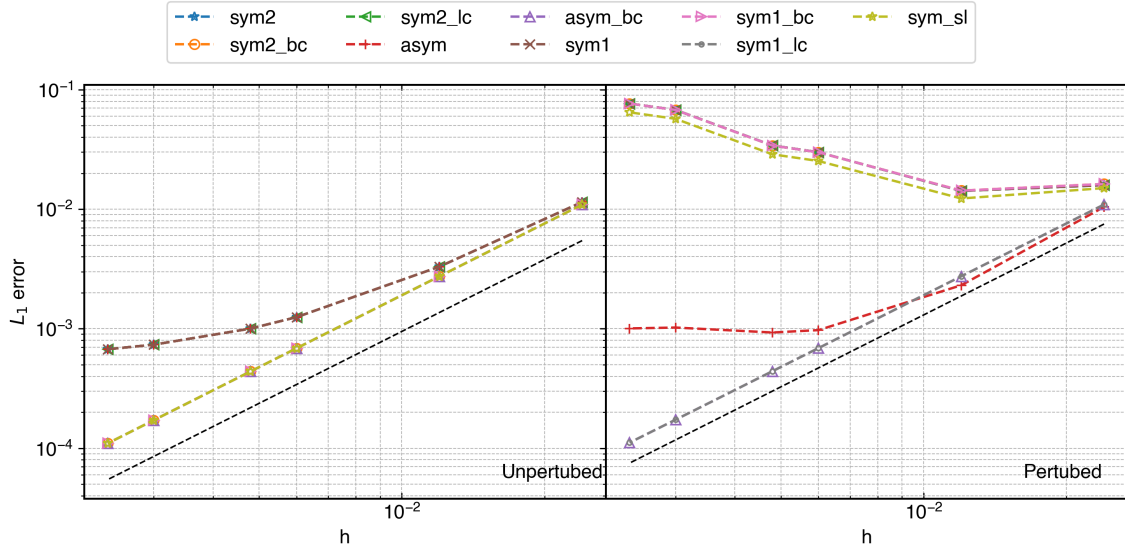


Figure 2.10 : The rate of convergence in UP (left) and PP (right) domains for various pressure gradient listed in table 2.2. The dashed line shows the SOC rate. The _bc and _lc suffixes represent the corresponding form with Bonet correction and Liu correction, respectively. The sym_sl is the sym1 formulation with symmetrization of kernel proposed in Dilts (1999).

Name	$\frac{F_T}{F_{max}}$	T_r	L_1 error	Order
asym_bc	1.01e-02	1.97	1.11e-04	1.99
asym	3.10e-04	1.00	1.01e-03	0.98
sym1_bc	1.08e-02	1.87	7.65e-02	-0.75
sym1_lc	1.01e-02	2.35	1.11e-04	1.99
sym1	-2.53e-11	1.05	7.65e-02	-0.76
sym2_bc	1.10e-02	1.87	7.65e-02	-0.75
sym2_lc	-1.90e-11	2.43	7.65e-02	-0.76
sym2	-1.87e-11	1.01	7.65e-02	-0.76
sym_sl	-1.73e-11	2.50	6.46e-02	-0.72

Table 2.3 : The ratio $\frac{F_T}{F_{max}}$ showing the total force in the system due to the lack of conservation in the approximation, the time taken T_r relative to the `asym` formulation, the L_1 error for 500×500 particle in a PP domain, and the last column shows the order of convergence for all the methods listed in the first column. The formulations that show convergence is shown in red.

cretization error in the approximation (Fatehi et al. (2011), Quinlan et al. (2006)). We also observe that with the correction, the second term involving the p_i term is zero in a UP domain leading to the same expression for all the formulations.

In the case of the PP domain, we observe that both `sym1` and `sym2` and their corresponding `_bc` versions overlap. The symmetric formulations show an increase in the error in the approximation with increasing resolution as suggested by Fatehi et al. (2011). Furthermore, the Bonet correction does not correct the symmetric formulations. Clearly, the `asym` formulation shows better convergence, and the Bonet correction version shows SOC. Therefore, the Bonet correction can be applied when an asymmetric formulation is employed. Moreover, the Liu correction only corrects the symmetric form `sym1`, which suggests that the `sym2` cannot be corrected using traditional correction techniques. We discuss this behavior in section 2.3.1 in detail. Finally, the `sym_sl` method has a slightly lower error but loses SOC behavior due to the symmetrization of the kernel gradient. Frontiere et al. (2017) reported similar behavior.

We also compare the linear momentum conservation and time taken to evaluate the gradient for the case with 500×500 particles. As shown in Bonet et al. (1999), linear momentum is conserved when the total force $F_T = \sum_i F_i = 0$, where the sum is taken over all the particles and $F_i = \frac{\nabla p_i}{\rho_i}$. In table 2.3, we tabulate the ratio of total force to the maximum force $F_{max} = \max\{F_i \forall i \in N\}$, the time taken to evaluate the gradient scaled

by the minimum time taken out of all the methods referred as T_r , and the L_1 error with the order of convergence ¹, for all the formulations plotted in fig. 2.10. As expected, all the symmetric forms of approximation have zero total force. The asymmetric formulation has a very small total force. Clearly, the use of Bonet correction increases the total force and slows down the computation by a factor of 2, whereas the Liu correction makes it 2.4 times slower. The `sym_sl` formulation shows zero residual force as expected. Using the table 2.3, we can see that `asym_bc` and `sym1_lc` show SOC and have a very low total force which makes them a suitable candidate for a scheme with SOC. We simulate inviscid flows in section 2.4.4 to investigate the implication of these forces on conservation.

2.2.3 Comparison of $\nabla^2\mathbf{u}$ approximation

In this section, we compare various approximations for the Laplacian operator discussed in section 1.3.3 also listed in table 2.4. We refer to the symmetric formulations of Cleary et al. (1999) and Brookshaw (1985) as `Cleary`, and those of Adami et al. (2013) as `tvf`. These ensure that linear momentum is conserved. We also consider the coupled formulation used by Bonet et al. (1999) and Nugent et al. (2000) and refer to these as `coupled`. This formulation shows oscillations in the approximation when the initial condition is discontinuous. However, to remedy this, one can perform an accurate first-order approximation near the discontinuity and then perform this approximation as proposed by Biriukov et al. (2019). We consider the improved formulation proposed by Fatehi et al. (2011) referred to as `Fatehi`. Both `coupled` and `Fatehi` formulations are asymmetric. These formulations are performed in two stages. The first stage involves the computation of the velocity gradient for each particle before the computation of the Laplacian. We also consider the correction applied to each of these formulations. In the case of `Cleary`, `tvf`, and `coupled` methods, we use the standard Bonet and Liu corrections. However, in the case of `Fatehi`, we use the correction tensor proposed by Fatehi et al. (2011). We also consider the method proposed by Korzilius et al. (2017) that remedies the deficiencies in earlier approaches where the second derivative was employed. We refer to this formulation as `Korzilius`.

In fig. 2.11, we plot the rate of convergence for the various formulations discussed above in both UP and PP domains. In the UP domain, all the methods at least show zeroth-order convergence. All methods without corrections suffer from high discretization error that dominates at higher resolutions (see appendix A.4). When either Bonet or Liu corrections are employed, `Cleary`, `coupled`, `Fatehi_c`, and `Korzilius` methods

¹In this work, we report the order of convergence by fitting a linear regression line and finding its slope.

Name	Expression	Used in
Cleary_bc	$\sum_j 2(\mathbf{u}_i - \mathbf{u}_j) B_i \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} ^2} \omega_j$	
Cleary_lc	$\sum_j 2(\mathbf{u}_i - \mathbf{u}_j) L_i \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} ^2} \omega_j$	-
Cleary	$\sum_j 2(\mathbf{u}_i - \mathbf{u}_j) \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} ^2} \omega_j$	WCSPH
Fatehi_c	$F_i \sum_j 2\omega_j \left(\frac{(\mathbf{u}_i - \mathbf{u}_j)}{ \mathbf{x}_{ij} } - \frac{\mathbf{x}_{ij} \cdot \langle \nabla \mathbf{u} \rangle_i}{ \mathbf{x}_{ij} } \right) \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} }$	Fatehi et al. (2011)
Fatehi	$\sum_j 2\omega_j \left(\frac{(\mathbf{u}_i - \mathbf{u}_j)}{ \mathbf{x}_{ij} } - \frac{\mathbf{x}_{ij} \cdot \langle \nabla \mathbf{u} \rangle_i}{ \mathbf{x}_{ij} } \right) \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} }$	Fatehi et al. (2011)
Korzilius	$K_i \left(\sum_j (u_j - u_i) \tilde{\nabla}^2 W_{ij} \omega_j - \mathbf{x}_{ij} \cdot \langle \nabla \mathbf{u} \rangle_i \tilde{\nabla}^2 W_{ij} \omega_j \right)$	Korzilius et al. (2017)
coupled_bc	$\sum_j (\langle \nabla \mathbf{u} \rangle_j - \langle \nabla \mathbf{u} \rangle_i) \cdot B_i \nabla W_{ij} \omega_j$	-
coupled	$\sum_j (\langle \nabla \mathbf{u} \rangle_j - \langle \nabla \mathbf{u} \rangle_i) \cdot \nabla W_{ij} \omega_j$	Bonet et al. (1999)
tvf_bc	$\sum_j \frac{1}{m_i} (\omega_i^2 + \omega_j^2) (\mathbf{u}_i - \mathbf{u}_j) B_i \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} ^2}$	-
tvf_lc	$\sum_j \frac{1}{m_i} (\omega_i^2 + \omega_j^2) (\mathbf{u}_i - \mathbf{u}_j) L_i \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} ^2}$	-
tvf	$\sum_j \frac{1}{m_i} (\omega_i^2 + \omega_j^2) (\mathbf{u}_i - \mathbf{u}_j) \frac{\nabla W_{ij} \cdot \mathbf{x}_{ij}}{ \mathbf{x}_{ij} ^2}$	TVF, EDAC

Table 2.4 : The various approximations of $\nabla^2 \mathbf{u}$. The formulations with blank ‘‘Used in’’ columns are not used in any existing scheme.

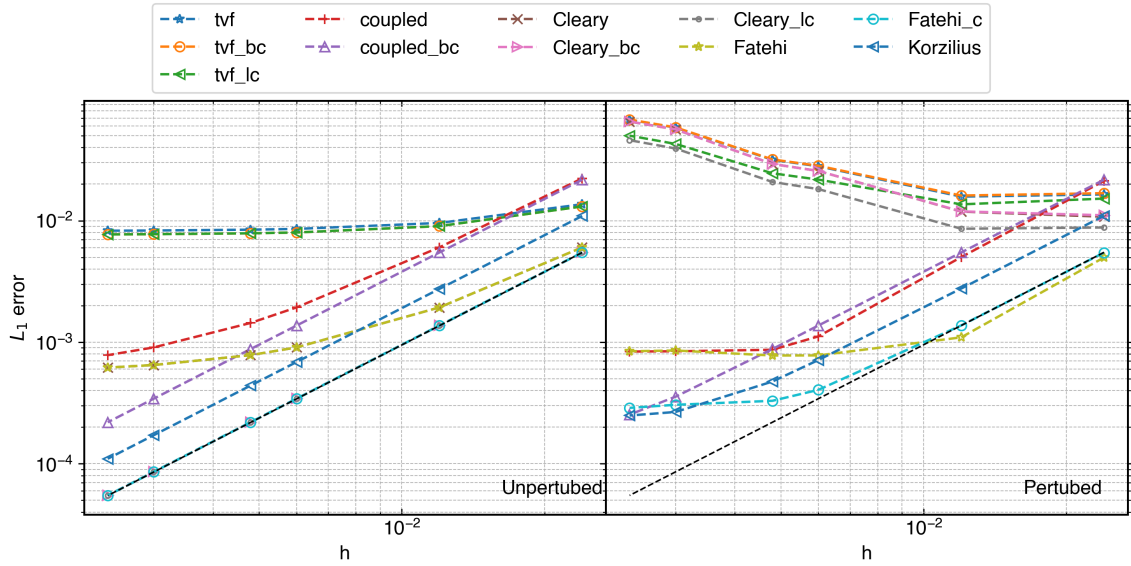


Figure 2.11 : The rate of convergence UP (left) and PP (right) domains for various approximations of the Laplacian operator in table 2.4. The dashed line shows the SOC rate. The suffixes `_bc` and `_lc` represent the corresponding form with the Bonet correction and Liu correction, respectively. The `Fatehi_c` refers to the `fatehi` formulation with the correction proposed by Fatehi et al. (2011) (see appendix A.3.4).

show SOC. The `coupled` method is approximately half an order less accurate as compared to `Cleary` and `Fatehi`. The accuracy of the `Korzilius` method is in between the

Name	$\frac{F_T}{F_{max}}$	T_r	L_1 error	Order
Cleary_bc	-1.28e+00	1.95	6.55e-02	-0.83
Cleary_lc	-2.15e-01	2.43	4.59e-02	-0.79
Cleary	-1.08e-10	1.12	6.54e-02	-0.84
Fatehi_c	1.69e+00	4.10	2.88e-04	1.30
Fatehi	1.30e+00	2.70	8.43e-04	0.68
Korzilius	1.61e+00	4.66	2.49e-04	1.70
coupled_bc	1.61e+00	3.05	2.54e-04	1.95
coupled	1.30e+00	2.59	8.38e-04	1.46
tvf_bc	-1.26e+00	2.07	6.80e-02	-0.66
tvf_lc	-2.16e-01	2.35	5.01e-02	-0.56
tvf	-1.06e-10	1.00	6.77e-02	-0.67

Table 2.5 : The ratio $\frac{F_T}{F_{max}}$ showing the total force in the system due to the lack of conservation of the approximation and the time taken, T_r relative to the tvf formulation, and L_1 error for 500×500 particle case in a PP domain. The last column shows the order of convergence for all the methods listed in the first column. The formulations that show convergence is shown in red.

coupled and Fatehi methods. The tvf method is very inaccurate as the discretization error increases due to the introduction of $\omega_i^2 + \omega_j^2$.

It is important to note that in the PP domain, the symmetric methods diverge due to the discretization error of $O\left(\frac{\tilde{d}_i}{h^2} \frac{\Delta x}{h}\right)$ (see table A.7). Only the coupled, Korzilius, and Fatehi methods show a positive convergence rate. On applying the corresponding correction, the coupled, Korzilius, and Fatehi methods improve. The accuracy for coupled, Korzilius, Fatehi is maintained, as observed in the case of the UP domain. However, at high resolution the quadrature error dominates.

In table 2.5, we tabulate the total force as a result of the approximation, the relative time taken for the approximation, and the error on a PP domain consisting of 500×500 particles with the order of convergence in the last column for each method plotted in fig. 2.11. We observe a similar increase in computational time due to the Bonet and Liu corrections, as seen in the case of gradient approximation. The coupled, Korzilius, and Fatehi formulations have even higher computational costs due to the additional step of velocity gradient computation. The Korzilius method requires additional time since the double derivative of the kernel is involved. The Fatehi_c method has an additional step where we compute the second-order tensor as shown in appendix A.3.4 for each particle

resulting in a further increase in computation time. We observe a similar increase in total force when an asymmetric version of the formulation is employed, as seen in the case of gradient approximation. Clearly, coupled, Korzilius, and Fatehi formulation results in an equal amount of total force resulting in a lack of conservation of linear momentum. In order to get a SOC approximation, we can use either of `coupled_bc`, `Korzilius`, or `Fatehi_c` formulations for viscous force estimation.

In the next section, we use the observations in section 2.2 to construct a SOC scheme.

2.3 The second-order convergent WCSPH scheme

We observe that many operator discretizations show second-order convergence. However, a convergent WCSPH implementation is still a grand challenge (Vacondio et al., 2020) in SPH literature. In this section, we propose a second-order convergent scheme and its variation. In the following sections, we point out some approaches which may make the scheme difficult to converge.

2.3.1 Considerations while applying kernel gradient correction

In section 2.2, we observed that some discretization do not converge even after applying the kernel gradient correction. In order to explain the behavior, we consider the first-order Taylor series approximation of a function f defined at \mathbf{x} about \mathbf{x}_i , given by

$$f(\mathbf{x}) = f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i) \cdot \nabla f(\mathbf{x}_i) + O(|\Delta x|^3). \quad (2.7)$$

Integrating both sides with $\nabla W(\mathbf{x} - \mathbf{x}_i)$ over the entire domain, we get

$$\begin{aligned} \int f(\mathbf{x}) \nabla W d\mathbf{x} &= \int f(\mathbf{x}_i) \nabla W d\mathbf{x} + \int (\mathbf{x} - \mathbf{x}_i) \cdot \nabla f(\mathbf{x}_i) \nabla W d\mathbf{x} \\ &= \int f(\mathbf{x}_i) \nabla W d\mathbf{x} + \int (\nabla W \otimes (\mathbf{x} - \mathbf{x}_i)) \nabla f(\mathbf{x}_i) d\mathbf{x}, \end{aligned} \quad (2.8)$$

ignoring the higher-order terms. Using a one-point quadrature approximation (Dilts, 1999), we get

$$\begin{aligned} \sum_j f_j \nabla W_{ij} \omega_j &= \sum_j f_i \nabla W_{ij} \omega_j + \sum_j \nabla W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \nabla f(x_i) \omega_j \\ \implies \nabla f(x_i) &= \sum_j (f_j - f_i) B_i \nabla W_{ij} \omega_j, \end{aligned} \quad (2.9)$$

where $B_i = \left(\sum_j \nabla W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \right)^{-1}$ is the correction matrix proposed by Bonet et al. (1999). Clearly, the first-order Taylor-series automatically suggests correction proposed

in Bonet et al. (1999) on an asym formulation. Equation (2.9) is $O(h^2)$ accurate (see table A.1).

On the other hand, the correction proposed by Liu et al. (2006), originates by convolving the Taylor series with $W(\mathbf{x} - \mathbf{x}_j)$ and $\nabla W(\mathbf{x} - \mathbf{x}_j)$, and solving all the equation simultaneously (see appendix A.3.2). For a constant field, this method ensures that we satisfy $\sum \tilde{W}_{ij}\omega_j = 1$ and $\sum \nabla \tilde{W}_{ij}\omega_j = 0$, where \tilde{W} is the corrected kernel. Therefore, with this correction in both sym1_1c and asym_1c formulations, the second term

$$p_i L_i \sum_j \nabla W_{ij} \omega_j,$$

becomes zero, and we get the SOC approximation. Whereas, in sym2_1c, the term

$$\frac{p_i}{\rho_i^2} L_i \sum_j \nabla W_{ij} m_j,$$

does not become zero. Thus even this correction fails to improve the approximation. Similarly, using the Taylor series expansion, Fatehi et al. (2011) derived a correction for the Laplacian operator. Therefore, the kernel gradient correction works with a certain form of the weight resulting in a linearly consistent formulation.

2.3.2 Considerations for the initial particle distribution

The particle distribution plays an important role in the error estimation of divergence approximation. In this section, we use first-order Taylor series approximation to obtain the error in divergence approximation as done in the previous section. We consider a two-dimensional velocity field. We write the error E_i , in the divergence evaluation as

$$E_i = \nabla \cdot \mathbf{u}_i - \sum_j (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j, \quad (2.10)$$

Using first-order Taylor-series expansion of \mathbf{u}_j about the point \mathbf{x}_i ,

$$\mathbf{u}_j = \mathbf{u}_i - (\mathbf{x}_{ij} \cdot \nabla) \mathbf{u}_i, \quad (2.11)$$

we write

$$\begin{aligned} E_i = & \left(1 - \sum_j x_{ij} \frac{\partial W_{ij}}{\partial x} \omega_j \right) \frac{\partial u_i}{\partial x} + \left(1 - \sum_j y_{ij} \frac{\partial W_{ij}}{\partial y} \omega_j \right) \frac{\partial v_i}{\partial y} \\ & - \sum_j y_{ij} \frac{\partial u_i}{\partial y} \frac{\partial W_{ij}}{\partial x} \omega_j - \sum_j x_{ij} \frac{\partial v_i}{\partial x} \frac{\partial W_{ij}}{\partial y} \omega_j. \end{aligned} \quad (2.12)$$

In the case of a UP domain, in eq. (2.12), the last two terms are exactly zero, and the coefficient of the first two terms are of equal magnitude. Furthermore, for a divergence-free velocity field, $\frac{\partial u_i}{\partial x} = -\frac{\partial v_i}{\partial y}$. Therefore, the overall error becomes zero. On the other

hand, in a PP domain, the last two terms are of equal magnitude and thus cancel, and the first two terms are different to the order 10^{-4} (see section 2.2.1) which becomes the leading error term. Thus, we always get an error of the order of 10^{-4} even after applying the Bonet correction. As far as we are aware, no known SPH discretizations can resolve this issue using a simple correction as done in the case of gradients. This is a possible avenue for future research.

2.3.3 Minimal requirements for a SOC scheme

In this section, we discuss the minimal requirement to obtain a SOC scheme for weakly compressible fluid flows. In the existing WCSPH scheme, the pressure gradient term is discretized using either `sym1` or `sym2` formulation. The value of ρ in the formulations is obtained using the continuity equation or from the particle density. Furthermore, this ρ is used to evaluate acceleration due to pressure gradient.

Let us consider the first method, where the continuity equation is used to determine the density of particles. Since the volume $\omega_j = m_j/\rho_j$. Therefore, integration volume will change according to the divergence of velocity. Hence, the integration volume ω is linked to the density, which is to the velocity field.

In the second method, where particle density ψ is used to determine density ρ . We note that the symmetric formulation used in all existing scheme to evaluate pressure gradient ensures a uniform distribution of particle, e.g. for `sym1`

$$\sum_j (p_j + p_i) \nabla W_{ij} \omega_j = \sum_j (p_j - p_i) \nabla W_{ij} \omega_j + 2p_i \sum_j \nabla W_{ij} \omega_j, \quad (2.13)$$

where the first term on the right is the zero-order approximation and the second term is a kind of regularization force. Therefore, the regularization force is forcing particles to be equispaced. However, an uneven particle distribution is required to identify high and low pressure region while using the particle density. In view of this relation between the particle density and the fluid density, and the identified convergent formulations from previous sections, we propose the following requirements to construct a SOC scheme.

1. We propose to use density as a transport property carried with the particle like velocity and pressure. We rewrite the governing equation, given by

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{u}, \quad (2.14a)$$

$$\frac{d\mathbf{u}}{dt} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}, \quad (2.14b)$$

where ρ , \mathbf{u} , and p are transport properties. Furthermore, the pressure is updated using

$$p_i = \rho_o c_o^2 \left(\left(\frac{\rho_i}{\rho_o} \right)^7 - 1 \right), \quad (2.15)$$

where ρ_o is the reference density. Equation (2.15) is the equation of state for water proposed by Batchelor (1967) that models the weakly-compressible nature accurately.

2. In order to evaluate the integration volume ω , we use the particle density ψ . Therefore, the integration volume is computed as

$$\omega_i = \frac{m_i}{\psi_i}, \quad (2.16)$$

where

$$\psi_i = \sum_j m_j W_{ij}. \quad (2.17)$$

Thus, we can approximate the density ρ using the standard SPH approximation given by

$$\rho_i = \sum_j \rho_j W_{ij} \omega_j. \quad (2.18)$$

3. We propose to employ the SOC approximations as discussed in section 2.2. In table 2.6, we list all the discretizations that we can employ to obtain a SOC WCSPH scheme.

Operators	Possible discretization for SOC
Gradient	asym_bc, sym1_lc
Divergence	div_c
Laplacian	coupled_c, Fatehi_c, Korzilius

Table 2.6 : The operators and their discretization suitable for a SOC scheme (For details, refer section 2.2).

4. Since we use an asymmetric form of the pressure gradient approximation, particles tend to clump together due to the absence of a redistributing background pressure (Sun et al., 2018). Therefore, we propose to use the iterative particle shifting proposed by Huang et al. (2019) after every few iterations to redistribute the particles.

Using the above four requirements, we propose a SOC WCSPH scheme in the next section.

A new SOC WCSPH scheme

In this scheme, we discretize the RHS in eq. (2.14) using SOC methods shown in table 2.6. We use `div_c` method to discretize the continuity equation given by

$$\frac{d\rho}{dt} = -\rho_i \sum_j (\mathbf{u}_j - \mathbf{u}_i) \cdot B_i \nabla W_{ij} \omega_j, \quad (2.19)$$

and in the momentum equation, we discretize the pressure gradient term using `asym_c` and the viscous term using `coupled_c`, given by

$$\frac{d\mathbf{u}}{dt} = - \sum_j \frac{(p_j - p_i)}{\rho_i} B_i \nabla W_{ij} \omega_j + \nu_i \sum_j (\langle \nabla \mathbf{u} \rangle_j - \langle \nabla \mathbf{u} \rangle_i) B_i \nabla W_{ij} \omega_j. \quad (2.20)$$

We obtain pressure using eq. (2.15). We note that the linear equation of state in eq. (A.36) works equally well. We evolve the particles in time using a Runge-Kutta 2nd order integrator given by

$$\begin{aligned} \phi^{n+1/2}(\mathbf{x}) &= \phi^n(\mathbf{x}) + \frac{1}{2} \Delta t \frac{d\phi^n(\mathbf{x})}{dt}, \\ \phi^{n+1}(\mathbf{x}) &= \phi^n(\mathbf{x}) + \Delta t \left(\frac{d\phi^{n+1/2}(\mathbf{x} + \frac{1}{2} \Delta t \mathbf{u})}{dt} \right), \end{aligned} \quad (2.21)$$

where Δt is the time step computed using eq. (A.54), and ϕ are all the transport properties like ρ and \mathbf{u} .

After every 10 timesteps, we perform particle regularization using IPST. We compute the shifting vector for the m^{th} iteration (of the shifting iterations) using

$$\delta \mathbf{x}_i^m = h_i \sum_j \mathbf{e}_{ij} W_{ij} \omega_j, \quad (2.22)$$

where $\mathbf{e}_{ij} = \mathbf{x}_{ij} / |\mathbf{x}_{ij}|$. The new particle position

$$\tilde{\mathbf{x}}_i^{m+1} = \mathbf{x}_i^m + \delta \mathbf{x}_i^m, \quad (2.23)$$

is computed. The particles are shifted until the criterion

$$|\max(\chi^m) - \chi_o| < \epsilon, \quad (2.24)$$

is satisfied up to a maximum of 10 iterations, where $\chi^m = h^2 \sum_j W_{ij}$, χ_o is the value for uniform distribution, and ϵ is an adjustable parameter. In order to keep the approximation of the particle $O(h^2)$ accurate, we update the particle properties after shifting by

$$\phi(\tilde{\mathbf{x}}_i) = \phi(\mathbf{x}_i) + (\tilde{\mathbf{x}}_i - \mathbf{x}_i) \cdot \nabla \phi(\mathbf{x}_i), \quad (2.25)$$

where $\tilde{\mathbf{x}}_i$ is the final position after iterative shifting, ϕ is the property to be updated, and $\nabla \phi(\mathbf{x}_i)$ is the gradient of the property on the last position computed with the Bonet correction. We refer to the scheme discussed above as L-IPST-C (Lagrangian with iterative PST and `coupled_c` viscosity formulation).

2.3.4 The effect of c_o on convergence

In the schemes discussed in the previous section, we impose artificial compressibility (AC) using the EOS, which is $O(M^2)$ as discussed in section 1.4.2. Chorin (1967) originally proposed this method to obtain steady-state solutions of an incompressible flow. Artificial compressibility with dual-time stepping may be used to achieve truly incompressible time-accurate results. We achieve the incompressibility limit when $c_o \rightarrow \infty$. Therefore, in order to increase the accuracy at higher resolution, a higher speed of sound must be used. We show the effect of the speed of sound on the convergence of the scheme in section 2.4.2.

2.3.5 Variations of the SOC scheme

In this section, we show that the L-IPST-C scheme presented in the section 2.3.3 can be easily converted into other models for improved accuracy and ease of calculation. We note that regardless of the model employed, the discretizations from table 2.6 must be used to achieve SOC.

In order to remove high-frequency oscillations, one could modify the continuity equation, given by

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{u} + D\nabla^2 \rho, \quad (2.26)$$

This modification corresponds to the δ -SPH scheme discussed in appendix A.5.2. However, we employ `coupled_c` formulation to discretize the density damping term. we could also use the linear equation of state proposed by Adami et al. (2013) to evaluate p , given by

$$p = c_o^2(\rho - \rho_o). \quad (2.27)$$

This EOS helps significantly reduce spurious pressure oscillation while simulating using traditional SPH schemes. Apart from the above minor modification, we propose the following major variation of the L-IPST-C scheme.

Variation in the particle shifting technique

In the L-IPST-C scheme, every 10 timestep, IPST is employed to achieve uniform distribution of particles. Lind et al. (2012) proposed a non-iterative method where particles are shifted by

$$\delta \mathbf{x}_i = 0.5 \frac{h^2}{\Delta t} \sum_j \left[1 + 0.2 \left(\frac{W_{ij}}{W(\Delta x)} \right)^4 \right] \nabla W_{ij} \omega_j, \quad (2.28)$$

where $W(\Delta x)$ is the kernel function value at the initial particle spacing (assuming uniform). We apply the above shifting after every iteration. Similar to IPST, the properties of

the particle are updated using eq. (2.25). We refer to this scheme as L-PST-C (Lagrangian with PST and coupled_c viscosity formulation).

SOC EDAC scheme

In the EDAC scheme, the pressure is evolved instead of density using the continuity equation. Taking the derivative of eq. (2.27), we get

$$\frac{dp}{dt} = c_o^2 \frac{d\rho}{dt}. \quad (2.29)$$

Similarly,

$$\nabla^2 p = c_o^2 \nabla^2 \rho. \quad (2.30)$$

Therefore, putting eq. (2.26) in eq. (2.29) and using eq. (2.30), we get

$$\begin{aligned} \frac{dp}{dt} &= -\rho c_o^2 \nabla \cdot \mathbf{u} + D c_o^2 \nabla^2 \rho \\ &= -\rho c_o^2 \nabla \cdot \mathbf{u} + D \nabla^2 p. \end{aligned} \quad (2.31)$$

We use `div_c` to discretize the divergence of velocity and `coupled_c` for the pressure damping term. Furthermore, we recover the artificial density ρ by inverting the linear equation of state, given by

$$\rho = \frac{p}{c_o^2} + \rho_o. \quad (2.32)$$

The momentum equation is discretized as in the original L-IPST-C scheme and IPST is used to regularize the particle distribution. We refer to this method as PE-IPST-C (pressure evolution with IPST and `coupled_c` viscosity formulation).

Remeshing in place of PST

The PST is used to regularize the particle distribution. Hieber et al. (2008) demonstrated the use of remeshing to reinitialize the particle position. The remeshing is performed using the M_4 kernel given by

$$M_4(q) = \begin{cases} 1 - \frac{5q^2}{2} + \frac{3q^3}{2} & 0 \leq q < 1, \\ \frac{(1-q)(2-q)^2}{2} & 1 \leq q < 2, \\ 0 & q \geq 2. \end{cases} \quad (2.33)$$

The property ϕ on the regular grid is computed using

$$\phi(\tilde{\mathbf{x}}_i) = \frac{\sum \phi(\mathbf{x}_j) M_4(|\tilde{\mathbf{x}}_i - \mathbf{x}_j|, h)}{\sum M_4(|\tilde{\mathbf{x}}_i - \mathbf{x}_j|, h)}, \quad (2.34)$$

where $\tilde{\mathbf{x}}$ are points on a regular Cartesian mesh. The remeshing procedure can be performed every few steps; however, we perform remeshing after every timestep. We use the coupled formulation for viscosity. We refer to this method as L-RR-C (Lagrangian with remeshing regularization and `coupled_c` viscosity formulation).

Replacing PST with transport velocity formulation

In the L-IPST-C scheme, the particle regularization is performed after the timestep. However, particle regularization can be included in the momentum equation as a regularization force. The regularization force is employed in TVF, δ^+ SPH, and ALE-SPH formulations as discussed in appendix A.5. Thus the particles are advected using the transport velocity, $\tilde{\mathbf{u}} = \mathbf{u} + \delta\mathbf{u}$, where $\delta\mathbf{u}$ is the shifting velocity, and the displacement is given by

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t(\mathbf{u}_i + \delta\mathbf{u}_i). \quad (2.35)$$

The new continuity and momentum equations are given by

$$\begin{aligned} \frac{\tilde{d}\rho}{dt} &= -\rho\nabla \cdot \mathbf{u} + D\nabla^2\rho + \delta\mathbf{u} \cdot \nabla\rho, \\ \frac{\tilde{d}\mathbf{u}}{dt} &= -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{u} + \delta\mathbf{u} \cdot \nabla\mathbf{u}, \end{aligned} \quad (2.36)$$

where $\frac{\tilde{d}(\cdot)}{dt} = \frac{\partial(\cdot)}{\partial t} + \tilde{\mathbf{u}} \cdot \nabla(\cdot)$. In this method, we employ SOC approximations mentioned in table 2.6 along with correction proposed in appendix A.6. The shifting velocity is computed as

$$\delta\mathbf{u}_i = -M(2h)c_o \sum_j \left[1 + 0.2 \left(\frac{W_{ij}}{W(\Delta x)} \right)^4 \right] \nabla W_{ij} \omega_j. \quad (2.37)$$

We refer to this scheme as TV-C (Transport velocity and `coupled_c` viscosity formulation).

SOC Eulerian scheme

The Eulerian method solves the equation of motion on a stationary grid. In the Eulerian description, the continuity equation is written as

$$\frac{\partial\rho}{\partial t} = -\rho\nabla \cdot \mathbf{u} - \mathbf{u} \cdot \nabla\rho. \quad (2.38)$$

Since the artificial fluid density ρ is not the same as the particle density ψ , we do not ignore the last term; this is unlike what is done by Nasar et al. (2019). The momentum equation is written as

$$\frac{\partial\mathbf{u}}{\partial t} = -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{u} - \mathbf{u} \cdot \nabla\mathbf{u}. \quad (2.39)$$

We discretize all the terms using SOC operators listed in table 2.6.

These set of equations can be derived using the TV-C method by setting $\delta\mathbf{u}_i = -\mathbf{u}_i$. This substitution makes the transport velocity in eq. (2.35) equal to zero; thus the particle does not move. The modified equation on setting $\delta\mathbf{u}_i = -\mathbf{u}_i$ in eq. (2.36), we get

$$\begin{aligned}\frac{\partial\rho}{\partial t} &= -\rho\nabla\cdot\mathbf{u} + D\nabla^2\rho - \mathbf{u}\cdot\nabla\rho, \\ \frac{\partial\mathbf{u}}{\partial t} &= -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{u} - \mathbf{u}\cdot\nabla\mathbf{u}.\end{aligned}\tag{2.40}$$

Therefore, we recover the governing equation for the Eulerian method. We refer to this method as E-C (Eulerian and coupled_c viscosity formulation).

Similarly, the scheme where the viscous term is discretized using Fatehi_c and Korzilius formulation are referred to as L-IPST-F and L-IPST-K, respectively.

2.4 Results and discussions

In this section, we compare the solution obtained from different schemes for the Taylor-Green, Gresho-Chan vortex, and incompressible shear layer problems. We first compare the L_1 error in velocity, pressure, and linear and angular momentum conservation of the L-IPST-C with existing schemes. In order to observe the effect of c_o on the convergence, we solve the Taylor-Green problem with different speeds of sound using the L-IPST-C and L-IPST-F schemes. For the highest value of $c_o = 80m/s$, we compare the results using different variations of the SOC schemes. In order to observe the conservation property, we compare the solutions for inviscid problems viz. incompressible shear layer and Gresho-Chan vortex using existing schemes as well as the SOC schemes. Furthermore, we compare the SOC and existing schemes for long-time simulations for all the test cases. Finally, we compute the cost of computation versus accuracy for all the schemes.

We implement the schemes using the open source PySPH (Ramachandran et al., 2021) framework and automate the generation of all the figures presented in this manuscript using the automan framework (Ramachandran, 2018).

2.4.1 Comparison with existing SPH schemes

In this section, we compare the following schemes:

1. The Transport Velocity Formulation (TVF) proposed by Adami et al. (2013) (see appendix A.5.3).
2. The Entropically Damped Artificial Compressibility (EDAC) SPH formulation proposed by Ramachandran et al. (2019) (see appendix A.5.4).

3. The improved δ -SPH formulation δ^+ SPH proposed by Sun et al. (2019) (see appendix A.5.5).
4. The Eulerian WCSPH (EWCSPPH) scheme proposed by Nasar et al. (2019) (see appendix A.5.6).
5. The L-IPST-C formulation discussed in section 2.3.3.

In order to compare these schemes, we consider the Taylor-Green vortex problem. We choose this problem since it is periodic, has no solid boundaries, and admits an exact solution². The solution to the Taylor-Green problem is given by

$$\begin{aligned}
 u &= -Ue^{bt} \cos(2\pi x) \sin(2\pi y), \\
 v &= Ue^{bt} \sin(2\pi x) \cos(2\pi y), \\
 p &= -0.25U^2 e^{2bt} (\cos(4\pi x) + \cos(4\pi y)),
 \end{aligned} \tag{2.41}$$

where $b = -8\pi^2/Re$, where Re is the Reynolds number of the flow. We consider $Re = 100$ and $U = 1m/s$. For the Lagrangian schemes, we consider a perturbed periodic (PP) arrangement of particles shown in fig. 2.1 for different resolutions. At $t = 0$ we initialize the pressure p and velocity (u, v) using eq. (2.41) for all the schemes. Since the fluid density ρ is a function of pressure, we initialize density inverting eq. (2.15). In the case of the EWCSPPH scheme, we consider an unperturbed periodic (UP) arrangement of particles and initialize the ρ using the prescribed pressure. We compute the L_1 error in pressure and velocity by

$$L_1(f, h) = \sum_j \sum_i \frac{|f(\mathbf{x}_i, t_j) - f_o(\mathbf{x}_i, t_j)|}{N} \Delta t, \tag{2.42}$$

where $h = h_{\Delta x} \Delta x$ is the smoothing length of the kernel, Δt is the timestep, N is the total number of particles in the domain, f is either pressure or velocity, and f_o is the exact value obtained using eq. (2.41). The particle spacing Δx is set according to the resolution. We consider resolutions of 50×50 to 500×500 particles in a $1m \times 1m$ periodic domain. In order to isolate the effect of spatial approximations on the convergence, we set the timestep $\Delta t = 0.3h/(U + c_o)$, where $h = 1.2/500m$ is set corresponding to highest resolution, $c_o = 10U$, for all the simulations. We run all the simulations for 1 timestep and observe convergence. We choose one timestep since most of the schemes considered diverge.

In fig. 2.12, we plot the L_1 error evaluated using eq. (2.42) for pressure and velocity in the domain for different schemes. Clearly, none of the schemes show convergence in pressure. This lack of convergence is present because the initial velocity is divergence-free, so there is no change in density and, thereby, pressure. We observe that the EDAC,

²We consider boundaries in chapter 6.

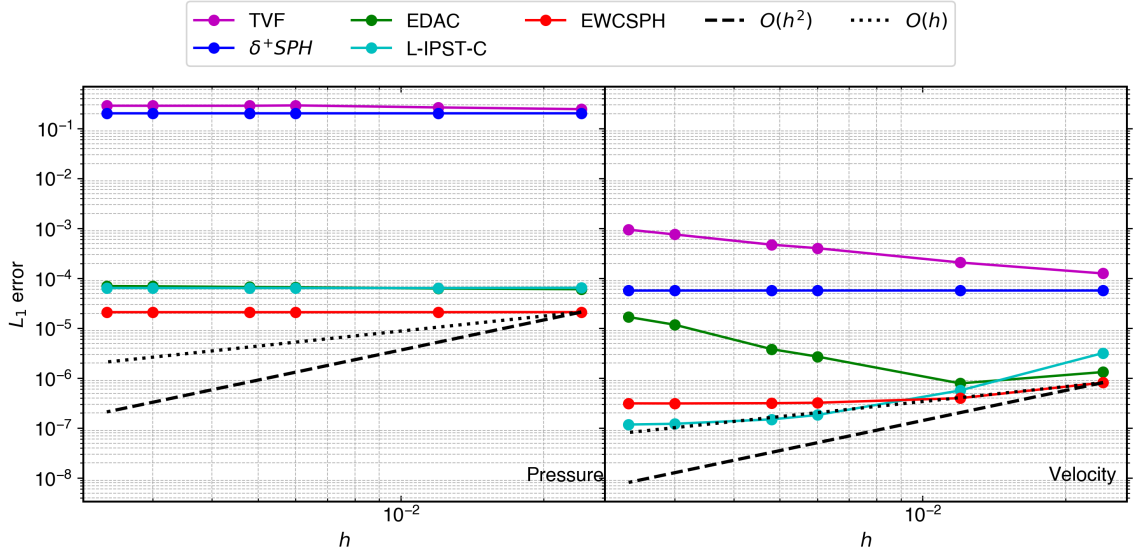


Figure 2.12 : Convergence of L_1 error in pressure (left) and velocity (right) with the change in resolution. $Re = 100$, $c_o = 10m/s$, $\Delta t = 6.54 \times 10^{-5} s$ and only 1 timestep taken.

EWCSPH, and L-IPST-C schemes are almost four orders more accurate than the TVF and δ^+ SPH schemes. In the case of both TVF and δ^+ SPH schemes, we link the pressure with particle density ψ , which is a function of the particle configuration. Since the particle positions are a result of the particle shifting, and therefore, the pressure is incorrectly captured. On the other hand, the other schemes either use a pressure evolution equation (EDAC) or a fluid density to evaluate pressure. In the case of the EWCSPH scheme, we initialize density using the pressure values in eq. (2.15) which results in better accuracy.

The L_1 error in velocity diverges in the case of the TVF and EDAC schemes since these use a symmetric form of type sym2 in table 2.2 to discretize the momentum equation. Whereas, in the case of the δ^+ SPH scheme, sym1 type of discretization is employed, leading to less errors. Moreover, the δ^+ SPH scheme uses a consistent formulation, and both TVF and EDAC schemes are inconsistent when the shifting (transport) velocity is added to the momentum equation Sun et al. (2019). The EWCSPH and L-IPST-C formulations show convergence (not second-order) as expected. We observe that in the velocity convergence, a constant leading error term dominates, resulting in flattening at higher resolutions. Since we use accurate second-order formulations in L-IPST-C and EWCSPH³ formulations, the only equation which is not converging with the resolution is the equation of state.

In this section, we have focused on highlighting the effect of using fluid density ρ different than the particle (or numerical) density ψ . The use of density as a transport

³It is second-order accurate since a uniform stationary grid is used.

Name	$\frac{F_r}{F_{max}}$	T_r	$L_1(\mathbf{u})(O)$	$L_1(p)(O)$
L-IPST-C	1.34e-05	3.36	1.18e-07(1.42)	6.41e-05(0.00)
EWCSPPH	7.03e-15	2.25	3.13e-07(0.38)	2.10e-05(0.00)
EDAC	1.88e-07	1.32	1.68e-05(-1.24)	7.00e-05(-0.07)
δ^+SPH	2.19e-05	1.49	5.69e-05(0.00)	2.03e-01(0.00)
TVF	3.70e-16	1.00	9.45e-04(-0.88)	2.88e-01(-0.07)

Table 2.7 : Table showing total force w.r.t. the maximum force in the domain and the time taken for 1 iteration w.r.t. the TVF scheme for all the schemes. The last two columns show L_1 error in velocity and pressure at 500×500 resolution with the order of convergence in the brackets.

property allows for a superior convergence rate and independence of density from particle positions. In contrast to this, the use of numerical density as a function of particle position is consistent with the volume used for the SPH approximation. In TVF, δ^+SPH , EDAC, and EWCSPPH schemes, there is no such distinction, i.e. $\rho = \psi$, and this density is used to compute numerical volume $\omega_j = m_j/\rho_j = m_j/\psi_j$. The poor convergence for these schemes shows that it is important to treat the fluid and numerical densities differently.

We also compare the linear momentum conservation and time taken to evaluate the accelerations for the case with 500×500 particles. As shown in Bonet et al. (1999), linear momentum is conserved when the total force, $\sum_i F_i = 0$, where the sum is taken over all the particles and $F_i = \frac{\nabla p_i}{\rho_i} + \nu \nabla^2 \mathbf{u}_i$. In the table 2.7, we tabulate the total force and the time taken by the scheme for one timestep with the errors and order of convergence in pressure and velocity for the 500×500 resolution case. It is clear that the TVF and EWCSPPH schemes conserve linear momentum, and the TVF scheme takes the least amount of time. The EDAC and the δ^+SPH scheme does not conserve linear momentum exactly. In the case of the EDAC scheme, the use of average pressure in the pressure gradient evaluation results in a lack of conservation. Whereas, in the case of δ^+SPH , the asymmetry of the shifting velocity divergence causes a lack of conservation. The L-IPST-C scheme is known to be non-conservative; however, the value is comparable to other schemes. The time taken by the L-IPST-C scheme is significantly higher due to the evaluation of correction matrices.

2.4.2 Convergence with varying speed of sound

In this section, we compare the convergence of the L-IPST-F and L-IPST-C schemes with change in AC parameter, i.e. the speed of sound c_o . We consider the Taylor-Green

problem; however, we run the simulation for $t = 0.5s$. In fig. 2.13, we plot the L_1 error in the pressure and velocity for both schemes and different c_o . We observe that both L-IPST-C and L-IPST-F methods are significantly affected by the change in c_o value, as expected. In the case of pressure, with the increase in the c_o value, the error in the lower resolutions increases; however, the convergence is monotonic. Clearly, we attain the increase in the order of convergence in case of the pressure due to increased error scales at lower resolutions. The increase in error is attributed to the inability of the SPH operators to correctly capture a divergence-free velocity field as discussed in section 2.3.2. However, on looking at the velocity convergence, both schemes attain SOC even at higher resolutions. As observed in the case of the Laplace operator comparison in 2.2.3, the use of Fatehi_c discretization used in L-IPST-F scheme offers better accuracy than the coupled_c discretization.

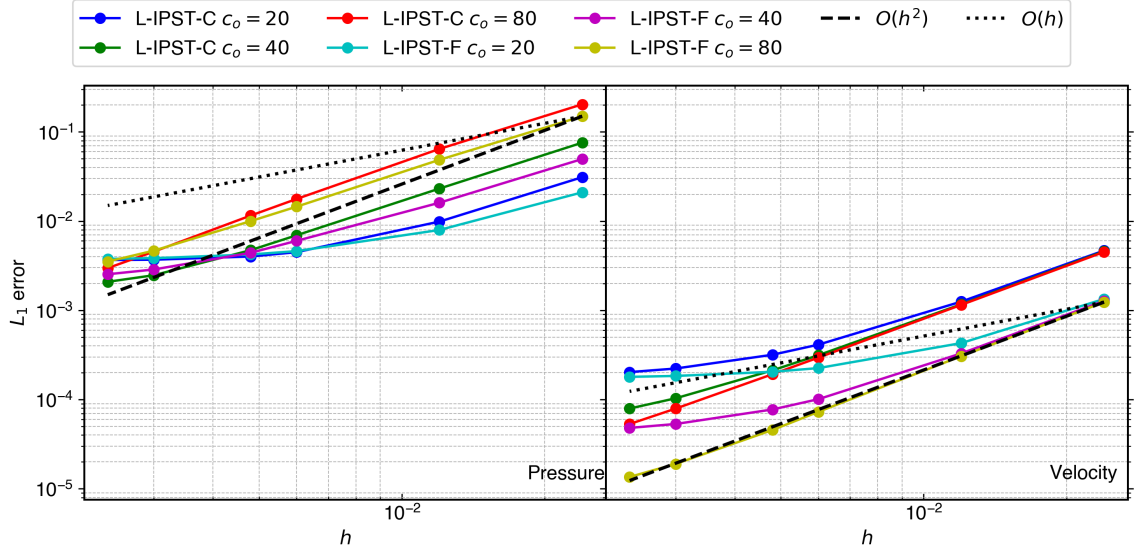


Figure 2.13 : Convergence rates for pressure (left) and velocity (right). The L-IPST-C and L-IPST-F methods are compared for different values of c_o .

In table 2.8, we tabulate the total force, relative time, and the L_1 error in pressure and velocity with the order of convergence for 500×500 particles. We observe that at a higher c_o value, the total force is higher compared to the simulation when c_o values are lower. From the table, we can see that the use of the L-IPST-F scheme offers better accuracy at the cost of the extra time taken. However, the order of improvement in the case of pressure is small. We also note that one can choose to use lower values of c_o at lower resolutions and increase the value as the resolution increases to get the same rate of convergence in velocity and better accuracy in pressure.

Name	$\frac{F_T}{F_{max}}$	T_r	$L_1(\mathbf{u})(O)$	$L_1(p)(O)$
L-IPST-C $c_o = 20$	7.13e-05	1.00	2.03e-04(1.38)	3.66e-03(0.93)
L-IPST-C $c_o = 40$	9.31e-05	2.15	7.94e-05(1.78)	2.09e-03(1.60)
L-IPST-C $c_o = 80$	1.44e-04	3.76	5.32e-05(1.93)	2.98e-03(1.85)
L-IPST-F $c_o = 20$	7.11e-05	1.23	1.80e-04(0.85)	3.77e-03(0.73)
L-IPST-F $c_o = 40$	5.99e-05	2.84	4.81e-05(1.44)	2.54e-03(1.31)
L-IPST-F $c_o = 80$	1.66e-04	5.46	1.36e-05(1.98)	3.52e-03(1.65)

Table 2.8 : Comparison of the total force, time taken relative to the L-IPST-C with $c_o = 20m/s$, L_1 error in velocity and pressure at 500×500 resolution with order of convergence in the brackets for different values of c_o .

2.4.3 Comparison of SOC variants

In this section, We simulate the Taylor-Green problem using $c_o = 80m/s$ for a duration of $0.5s$ with different resolutions for schemes discussed in section 2.3.5. In addition, we study the performance of the EWCSPH scheme since it is computationally efficient and accurate. In fig. 2.14⁴, we plot the error in pressure and velocity for all the schemes. In table 2.9, we tabulate the total force, relative time, L_1 error in pressure and velocity at 500×500 resolution, and the order of convergence.

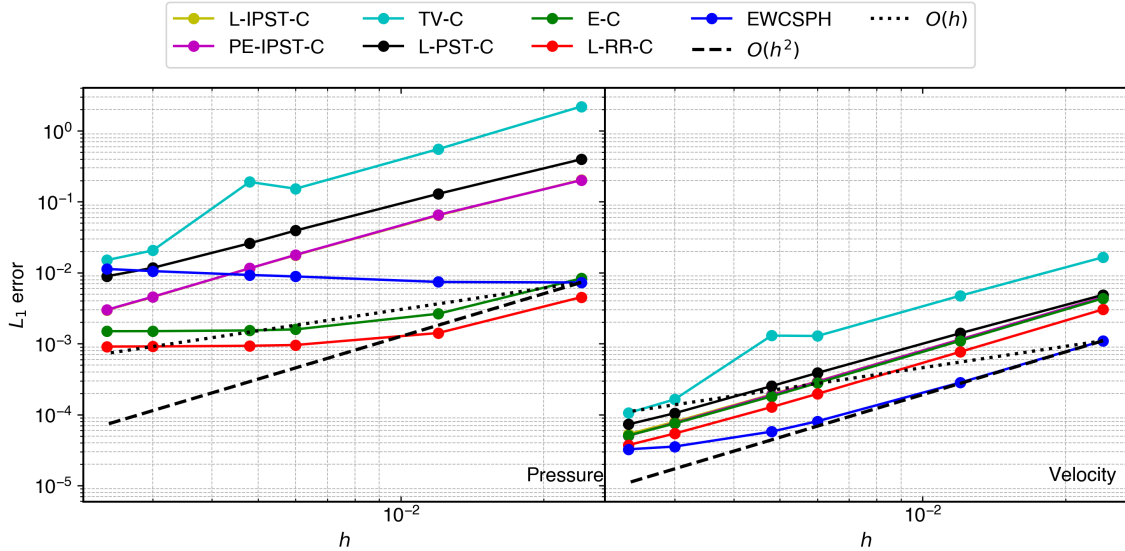


Figure 2.14 : Convergence rates for pressure (left) and velocity (right) of different variants of the SOC scheme.

⁴The cause of the non-monotonic behavior of the TV-C in the figure is not known, but the result is reproducible.

Name	$\frac{F_T}{F_{max}}$	T_r	$L_1(\mathbf{u})(O)$	$L_1(p)(O)$
E-C	8.13e-15	1.19	5.11e-05(1.94)	1.50e-03(0.71)
EWCSPPH	1.07e-14	1.00	3.25e-05(1.57)	1.13e-02(-0.20)
L-IPST-C	1.44e-04	1.53	5.32e-05(1.93)	2.98e-03(1.85)
L-PST-C	2.64e-06	2.02	7.31e-05(1.84)	8.93e-03(1.68)
L-RR-C	1.51e-18	1.41	3.74e-05(1.92)	9.11e-04(0.65)
PE-IPST-C	6.17e-05	1.65	5.05e-05(1.96)	3.01e-03(1.84)
TV-C	-5.01e-06	1.88	1.06e-04(2.18)	1.51e-02(2.14)

Table 2.9 : Comparison of the total force, time taken relative to L-IPST-C, L_1 error in velocity and pressure at 500×500 resolution with order of convergence in brackets for variation of SOC scheme with $c_o = 80m/s$.

The L-IPST-C and PE-IPST-C overlap in both pressure and velocity convergence plots, and these are both approximately second-order. Compared to L-IPST-C, the L-PST-C shows a lower convergence rate, and TV-C shows higher order of convergence, whereas E-C and L-RR-C show very poor convergence rates in pressure; however, the L-RR-C method shows very low errors in pressure. The EWCSPPH has a negative convergence rate in pressure. While the TV-C shows a high convergence rate, it has much larger errors than all the other schemes considered for both pressure and velocity. The E-C, L-IPST-C, L-RR-C, and PE-IPST-C shows high convergence rates in velocity, as expected. The L-PST-C shows a slightly high error and 1.84 convergence rate. The EWCSPPH shows a lower convergence rate of 1.57 but is the most accurate of all the schemes regarding the velocity error.

The TV-C scheme shows low accuracy since we perform the shifting using an additional term in the momentum equation compared to the PE-IPST-C and L-IPST-C. This decrease in accuracy is also visible in the case of velocity. The L-PST-C scheme show higher error suggesting that the non-iterative PST does not perform the required amount of regularization. Both L-RR-C and E-C are comparable and most accurate. These schemes have lower errors since the particles are fixed on a cartesian grid resulting in accurate computation of divergence as discussed in section 2.3.2. The pressure convergence flattens since it reaches the limit of accuracy possible with this value of $c_o = 80m/s$, and further accuracy may be seen by increasing this further.

Clearly, the total force in the case of E-C, L-RR-C, and EWCSPPH schemes is zero since we compute the acceleration on a uniform Cartesian grid of particles. However, the total force in other schemes is accurate to order 10^{-5} . The times taken shows that L-PST-

C is the highest since we apply the PST at every timestep. The TV-C involves many terms in the equations and therefore takes a lot of time. The E-C and EWCSPPH take the least time since they do not use a PST⁵. The L-RR-C, L-IPST-C, and PE-IPST-C take a similar amount of time.

2.4.4 Comparison of conservation errors

Thus far, we have looked at the convergence of the various schemes. In this section, we look at the existing schemes and the new schemes listed in table 2.9 from the perspective of conservation of linear and angular momentum. We solve the Gresho vortex and incompressible shear layer using all the schemes discussed.

The Gresho vortex

We consider the Gresho vortex problem (Gresho et al., 1990), which is an inviscid incompressible flow problem having the pressure and velocity fields given by

$$p(r) = \begin{cases} 12.5r^2 + 5 & 0 \leq r < 0.2, \\ 12.5r^2 - 20r + 4 \ln(5r) + 9 & 0.2 \leq r < 0.4, \\ 3 + 4 \ln(2) & 0.4 \leq r, \end{cases} \quad (2.43a)$$

$$u_\phi(r) = \begin{cases} 5r & 0 \leq r < 0.2, \\ 2 - 5r & 0.2 \leq r < 0.4, \\ 0 & 0.4 \leq r, \end{cases} \quad (2.43b)$$

$$(2.43c)$$

where $u_\phi(r)$ is the radial velocity.

We consider an unperturbed periodic domain of size 1×1 with the center at $(0, 0)$. We set the kinematic viscosity, $\nu = 0$, and the time step and other properties as done in the Taylor-Green problem. The problem is simulated until $t = 3s$. Since the problem is inviscid, we expect the scheme to retain the velocity and pressure field. We do not use artificial viscosity in the simulations for any of the schemes. However, we use density or pressure damping as given in eq. (2.26) or eq. (2.31) to reduce the pressure oscillations. Without this, the solution becomes unstable in a short amount of time. We perform the simulation of all the schemes listed in table 2.7 except the EWCSPPH scheme⁶. We note

⁵These methods can be made even faster since the neighbors need not be updated, and the correction matrices can also be computed once and saved.

⁶We discuss the failed simulations in appendix A.7.

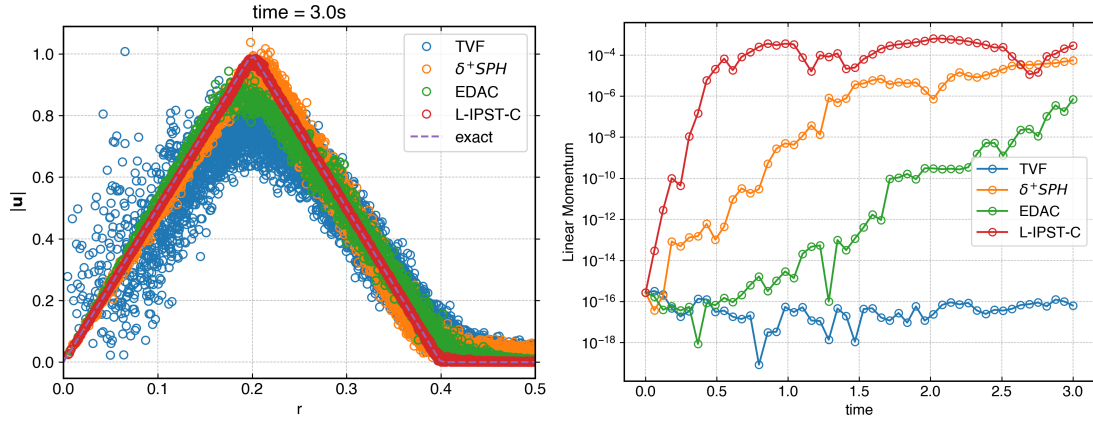


Figure 2.15 : The velocity of particles with the distance from the center of the vortex (left) and the x -component of the total linear momentum (right) for existing and L-IPST-C schemes.

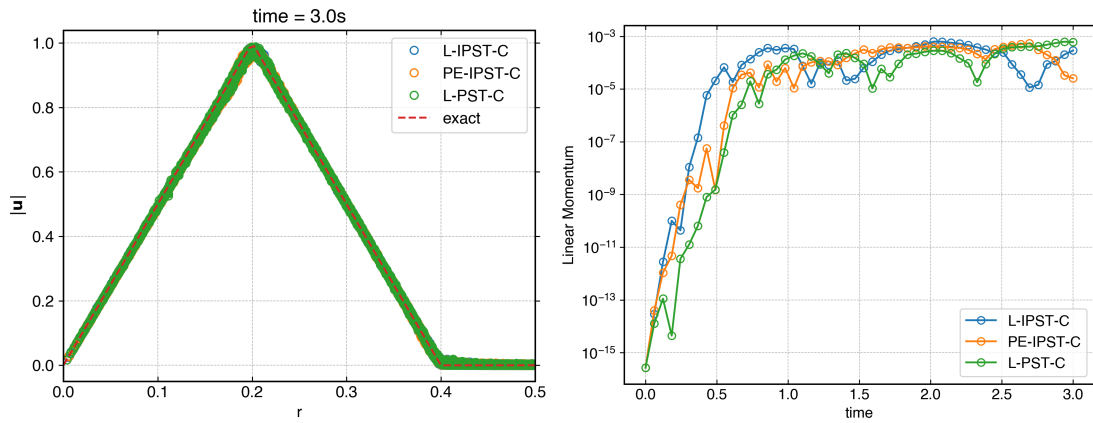


Figure 2.16 : The velocity of particles with the distance from the center of the vortex (left) and the x -component of the total linear momentum (right) for the variation of the SOC scheme.

that using an initial perturbed particle configuration results in very diffused results for all schemes except the L-IPST-C.

In fig. 2.15, we plot the velocity of the particles with the distance r from the center (on the left) and the x -component of the total linear momentum with time for a 100×100 particle simulation, for all the schemes. The L-IPST-C scheme retains the velocity profile very well. The δ^+SPH , EDAC, and TVF schemes show diffusion due to inaccuracy in the pressure gradient evaluation. Except for the TVF scheme, the rest show a finite increase in the momentum bounded at 10^{-4} . Clearly, approximate linear momentum conservation is sufficient to obtain accurate results in the case of weakly compressible flows.

We also perform the simulations with different versions of the SOC scheme listed in table 2.9⁷. In fig. 2.16, we plot the velocity with the distance from the center and the x -component of the linear momentum with time for 100×100 particle simulation. Clearly, all the schemes are accurate and approximately conserve linear momentum as expected.

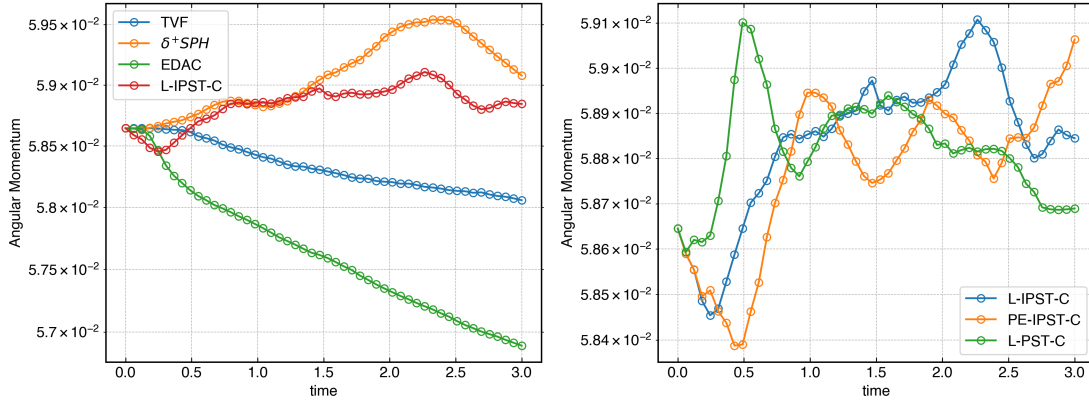


Figure 2.17 : The angular momentum variation with time for Gresho-Chan vortex for different schemes.

In fig. 2.17, we show the angular momentum variation with time for different schemes. None of the schemes conserve angular momentum, but for the SOC schemes, the variations are very small and at $O(5 \times 10^{-4})$.

The incompressible shear layer

The incompressible shear layer simulates the Kelvin-Helmholtz instability in an incompressible flow. This test case produces non-physical vortices for the schemes where the operators are under-resolved even when the scheme is convergent (Di et al., 2005). The initial condition for the velocity in the x direction is given by

$$u = \begin{cases} \tanh(30(y - 0.25)) & y \leq 0.5, \\ \tanh(30(0.75 - y)) & y > 0.5. \end{cases} \quad (2.44)$$

In order to begin the instability, a small velocity is given in y direction

$$v = 0.05 \sin(2\pi x). \quad (2.45)$$

We consider a small viscosity $\nu = 1/10000$. We simulate the problem using all the schemes listed in table 2.7. In fig. 2.18 and fig. 2.19, we plot the vorticity field for the

⁷The L-RR-C, TV-C, and E-C schemes fail to complete the simulation, and these are discussed in the appendix A.7.

schemes⁸ discussed in this work. Unlike the inviscid problem of the Gresho-Chan vortex, the scheme EWCSPH, TV-C, and E-C shows results matching other SOC schemes. In fig. 2.18, we observe that the TVF scheme and δ^+ SPH scheme show high-frequency oscillations, while the EDAC scheme is much better; However, it shows some undesirable contours surrounding the eye of the vortex in blue color.

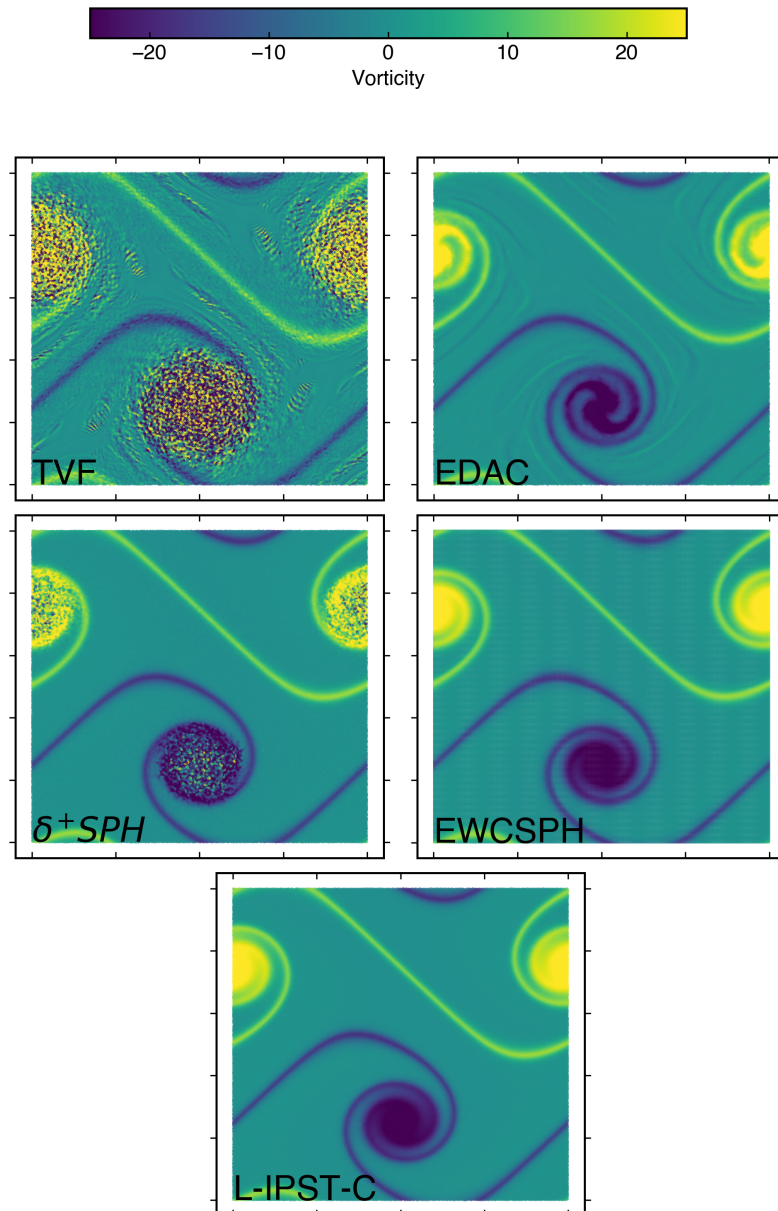


Figure 2.18 : Vorticity contour plot for 500×500 resolution for existing and L-IPST-C schemes.

⁸The L-RR-C method failed to run due to discontinuity in the initial velocity field.

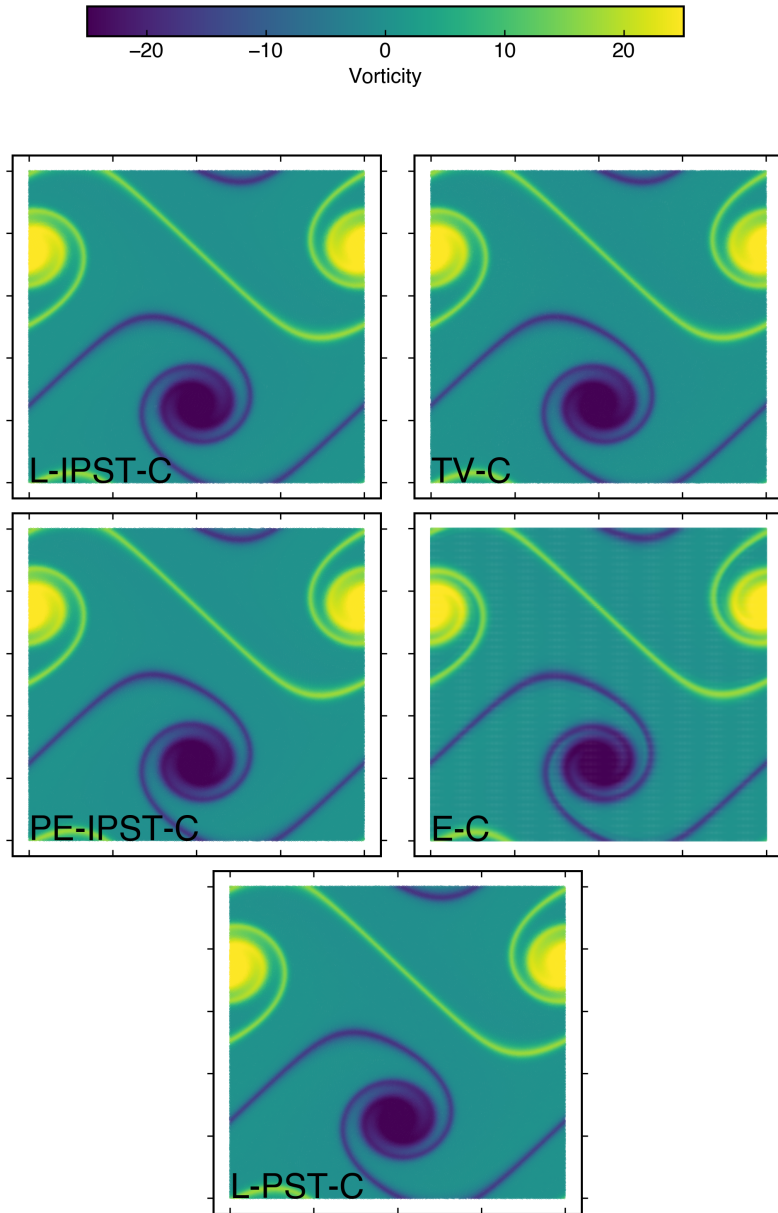


Figure 2.19 : Vorticity contour plot for 500×500 resolution for all the variation of the SOC scheme.

2.4.5 Long time simulations

In this section, we study the conservation for long-time simulations using the EDAC, TVF, and L-IPST-C schemes. We consider the Taylor-Green and Gresho-Chan with the same condition as before. We consider a UP particle distribution for all the schemes.

We simulate the Taylor-Green problem for $5s$ at $Re = 100$ compared to the final time of $0.5s$ in the previous simulations for all the schemes. In fig. 2.20, we plot the velocity damping and the kinetic energy of the flow as a function of time. The TVF scheme shows a significant deviation from the exact result, whereas the kinetic energy remains close

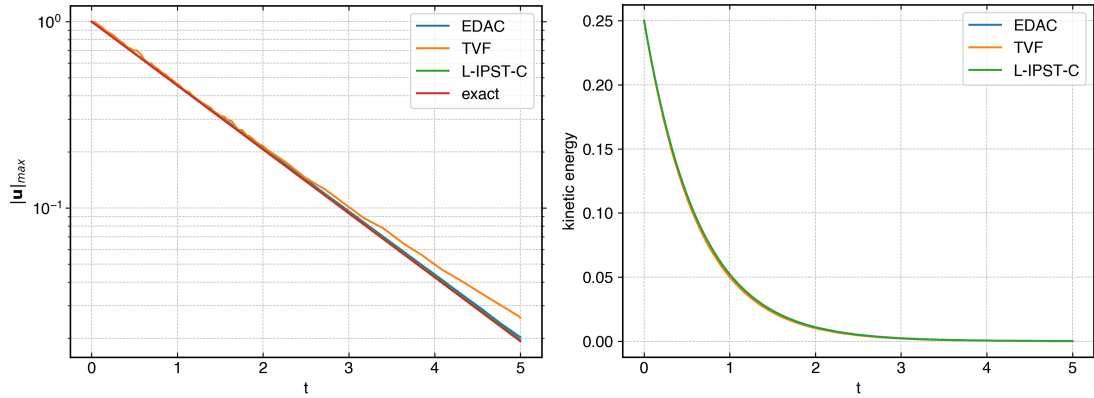


Figure 2.20 : The maximum velocity decay (left) and kinetic energy of the flow with respect time (right) for the Taylor-Green problem.

to other scheme solutions. We note that the TVF scheme conserves linear momentum exactly.

We next simulate the Gresho-Chan vortex problem for 7s compared to the final time of 3s in section 2.4.4. In fig. 2.21, we plot the velocity as a function of r and the linear and angular momentum with respect to time for all the schemes. We observe that the TVF scheme does not capture the physics of the problem however conserves linear momentum but does not conserve angular momentum. In the case of the EDAC scheme, the physics is captured better. The linear momentum is not conserved, the solution loses angular momentum by a small amount, and the peak of the velocity distribution is not captured accurately. The L-IPST-C scheme retains the velocity field, and both linear and angular momentums are approximately conserved. After 7 seconds, the L-IPST-C scheme is no longer stable, and the velocity field is not captured accurately.

These simulations suggest that even if a scheme is conservative like TVF, it may not produce accurate results. However, for a convergent scheme like the L-IPST-C, the results are accurate, and despite there being no exact conservation, approximate conservation is seen.

2.4.6 Cost of computation

In this section, we compare the cost of computation of all the schemes considered in this study. We simulate the Taylor-Green problem for 5000 timesteps with 50, 100, and 200 resolutions for all the schemes. We use an Intel(R) Xeon(R) CPU E5-2650 v3 processor and execute all the simulations in serial. In fig. 2.22, we plot the L_1 error in velocity computed using eq. (2.42) as a function of time taken for the simulation. Clearly, all the SOC schemes are close to each other in terms of errors. The E-C and EWCSPPH schemes take very little time and are very accurate; however, EWCSPPH is not convergent

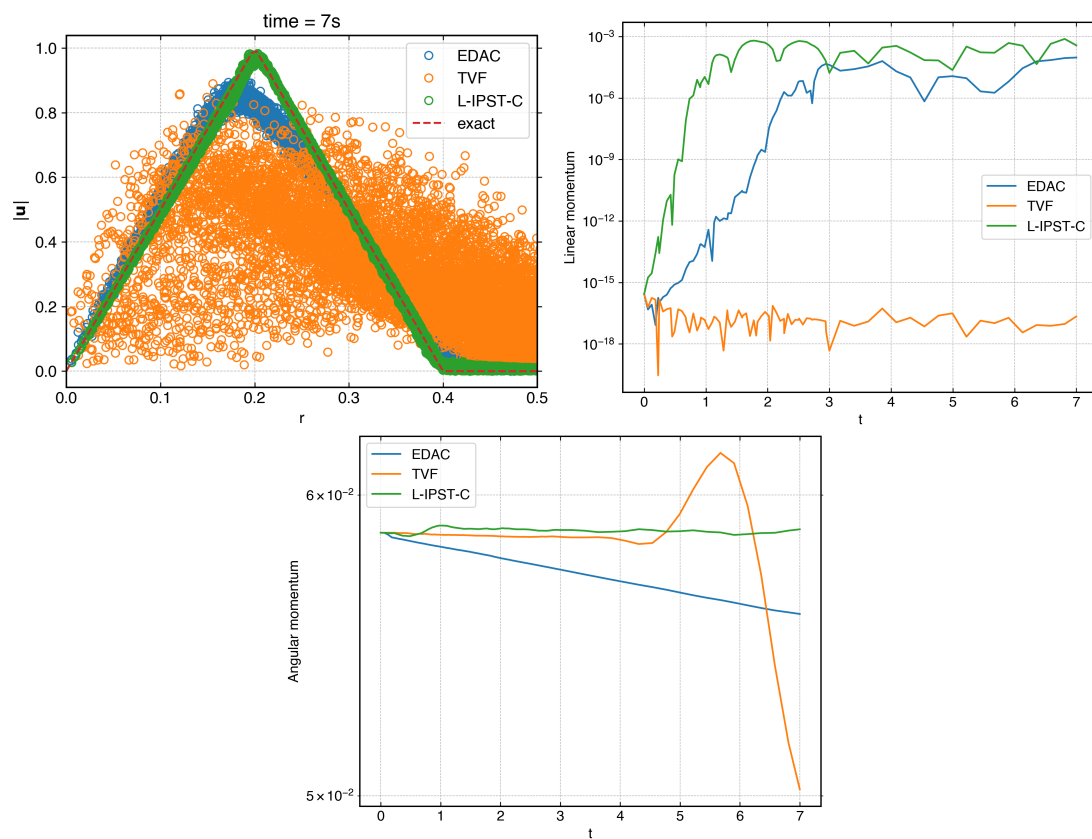


Figure 2.21 : The velocity of all the particles in the domain with distance from the center (top left), linear (top right) and angular (bottom) momentum with respect to time for Gresho-Chan vortex problem.

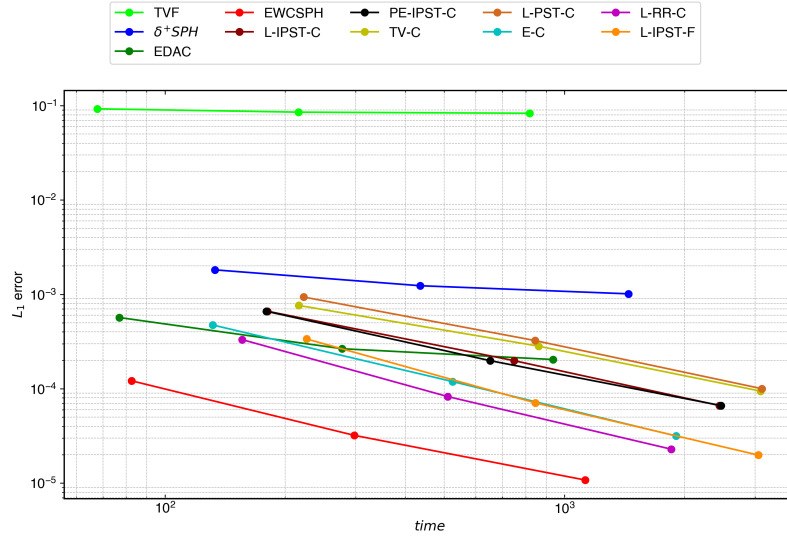


Figure 2.22 : The L_1 error in velocity with respect to the time taken to evaluate 5000 timesteps for all the schemes discussed in the previous sections.

in pressure as shown in section 2.4.1. The EDAC scheme has lower error comparable to the SOC schemes; however, its convergence rate reduces with an increase in resolution. We show that despite having higher time taken by the SOC schemes, they achieve higher accuracy with fewer particles. For some schemes, these accuracy levels are not achievable at all.

2.5 Summary

In this chapter, we have performed a numerical study of the accuracy and convergence of a variety of SPH schemes in the context of weakly-compressible fluids. Based on the numerical study performed in the previous sections, we summarize the key observations as follows:

1. Choice of smoothing kernel:

We first considered the SPH approximation of a function and its derivative using different kernels. All the kernels considered here show second-order convergence when the support radius is suitably chosen. The accuracy is marginally affected by the change in the type of kernel. The smoothing error of an SPH approximation scales as $O(h^2)$, and this necessitates that the smoothing length of the kernel is as small as possible. This implies that $h_{\Delta x}$ be small. As is well known, the discretization errors scale as $O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$ and this necessitates that the smoothing radius be larger. These two requirements are contradictory. We find that by using a modest $h = 1.2\Delta x$, along with the kernel corrections of Bonet et al. (1999) or Liu et al.

(2006) we are able to obtain close to second-order convergence for the kernels considered in this work. It holds up to a resolution of $L/\Delta x = 500$, where $L = 1m$, which appears to be among the highest resolutions we have seen in the literature concerning the convergence of SPH methods. In the literature, we find kernels like the cubic spline to demonstrate pairing instabilities (Dehnen et al., 2012). We can avoid this instability by using a particle shifting technique (PST).

2. Particle density and fluid density:

We recommend that one employ the fluid density ρ in the governing differential equation (see eq. (2.14)) as a property that convects with the particle. The approximation of the SPH operators should not be a function of a property of the fluid, i.e. density (as integration volume $\omega_i = m_i/\rho_i$ used in existing schemes, where ρ is evaluated from the continuity equation). We obtain the integration volume by eq. (2.16) where the mass and kernel support radius of particles is kept constant.

3. Choice of suitable operators:

The SPH approximation of operators like the gradient, divergence, and Laplacian must be chosen carefully. In this work, we recommend two methods for gradient approximation and three methods for viscous term approximation that ensure second-order convergence. The approximations which ensure pair-wise linear momentum conservation are always divergent. Furthermore, the widely used artificial compressibility assumption makes the scheme $O(M^2)$ accurate. We recommend using high c_o values or dual-time stepping criteria to achieve convergence.

4. SOC scheme and variations:

We demonstrate Eulerian as well as Lagrangian SPH schemes that are second-order convergent. We show that the Eulerian schemes capture the divergence accurately due to symmetry in the particle distribution resulting in better accuracy in pressure. We derive a pressure evolution equation using the continuity equation that resembles the EDAC SPH scheme in literature. We show that the PST step in the Lagrangian method can be replaced by a remeshing step which is another momentum conserving regularization. However, remeshing is not stable in the presence of jumps in the properties as observed in the case of the Gresho vortex (see appendix A.7) and incompressible shear layer. The PST step can be included in the momentum equation resulting in the δ^+ SPH scheme. From the δ^+ SPH method, one can obtain the Eulerian form of the WCSPH method by setting the shifting velocity to $-\mathbf{u}$. All these schemes are SOC when we use a second-order convergent

approximation for the operators. We show that even though the schemes are non-conservative in the absolute sense, approximate conservation also produces accurate results in the case of incompressible flows.

Thus, by a judicious choice of discretization, particle shifting, and a separation of the fluid and particle densities, we have shown that second-order convergence is possible using the SPH method for weakly-compressible flows.

In this chapter, we used the method of exact solution (MES), where for all the problems, an exact solution was available. Unfortunately, the method takes a significant amount of time due to the high artificial speed of sound to assess the convergence of a scheme. Also, the exact solutions of NS equations for three-dimensional domains are very rare. Furthermore, the boundary due to solid body, inlet or outlet, and free surface cannot be studied. In the next chapter, we discuss other methods used to obtain the rate of convergence of the SPH method.

Chapter 3

Verification techniques for WCSPH schemes

In the SPH literature, the accuracy of the SPH scheme is shown qualitatively by comparing it with established solvers. The rate of convergence is rarely reported. The computation of the rate of convergence of a computer code that solves physical phenomena comes under a broader concept called validation and verification. Salari et al. (2000) formally introduced the concept of validation and verification of a code used to solve partial differential equations, as discussed in section 1.5. Both verification and validation of a computer code are equally important. Verification of the accuracy and convergence of a code is found using exact solutions, solutions from existing solvers, experimental results, or manufactured solutions. The verification methods can also be used to identify bugs in the implementation. On the other hand, validation of the code ensures that the governing equations (or mathematical model) are appropriate for the physics of the problem and often involves comparison with experimental results.

In this chapter, we focus on the code verification techniques used in SPH. Salari et al. (2000) classified different methods for code verification as follows:

1. Trend test: An *expert judgment* is used to verify the solution obtained from the code. For example, the velocity of the vortex in a viscous periodic domain should diminish with time. If the code results in an increase of the velocity in the domain, then there is an error in the implementation.
2. Symmetry test: It is ensured that the solution obtained does not change if the domain is rotated or translated. For example, if we implement an inlet assuming the flow in the x direction, we will get an erroneous result on rotating the domain by 90 degree.

3. Comparison test: The solution obtained from the code is compared with the solutions from an established solver or experiment. The established solution and experiment do not always have a solution at all data points. Therefore, in this method, one is forced to evaluate the solution at certain prescribed locations.
4. Method of exact solution (MES): A problem is solved for which the exact solution is known. In the context of fluid dynamics, a closed-form solution is only available for simple problems.
5. Method of manufactured solutions (MMS): An artificial solution $f_o(x, y, z, t)$ is manufactured for the governing equation of the form $\mathcal{L}f = g$, where \mathcal{L} is differential operator, and $g = g(x, y, z, t)$ is an arbitrary forcing function. Since the f_o is not the solution, on substituting it in the governing equations, a residue $R = \mathcal{L}f_o - g$ is obtained. In the implementation, the residue R is added as a source term to recover the manufactured solution f_o . The MMS is widely used to verify codes in finite volume methods (Bond et al., 2007) and finite element methods (Waltz et al., 2014).

We first discuss the drawbacks of the currently used verification methods in the context of WCSPH schemes. The MES used is computationally expensive. Furthermore, the method could not pinpoint the root cause of the lack of convergence of a code. Moreover, these methods cannot be applied to determine the rate of convergence of boundary condition implementations. Feng et al. (2016) used the method of manufactured solutions (MMS) to verify their SPH implementation. However, the particles do not move; therefore, it is no different from a traditional application of MMS in mesh-based methods. We explore the well-established MMS in the context of WCSPH schemes. We, for the first time, show how one can apply the MMS to carefully study the accuracy of a modern WCSPH implementation.

We first obtain a suitable initial particle configuration to be used in the simulation. We then systematically develop an approach to construct a Manufactured Solution (MS) for established WCSPH schemes as well as the proposed second-order schemes. We show how this can be applied to any specified domain shape. We show how to apply the MMS in the context of both Eulerian and Lagrangian SPH schemes. We then demonstrate how the MMS can help to debug an implementation by deliberately changing one of the equations in the second-order convergent scheme and showing the effect of the change in the convergence plot. We demonstrate that the method can be used to study convergence at extreme resolutions and for three-dimensional cases.

3.1 Drawback of current verification methods

In this section, we compare solutions for the Taylor-Green vortex and lid-driven cavity problems, which are examples of MES and comparison test, respectively.

The Taylor-Green problem has an exact solution given by

$$\begin{aligned} u &= -Ue^{bt} \cos(2\pi x) \sin(2\pi y), \\ v &= Ue^{bt} \sin(2\pi x) \cos(2\pi y), \\ p &= -0.25U^2 e^{2bt} (\cos(4\pi x) + \cos(4\pi y)), \end{aligned} \quad (3.1)$$

where $b = -8\pi^2/Re$, where Re is the Reynolds number of the flow. We consider $Re = 100$ and $U = 1m/s$. We solve this problem for three different resolutions viz. 50×50 , 100×100 , and 200×200 for a two-dimensional domain of size $1m \times 1m$ for $2s$ using the L-IPST-C scheme. However, we discretize the pressure gradient using the formulation given by

$$\left\langle \frac{\nabla p}{\rho} \right\rangle_i = \sum_j \frac{(p_j + p_i)}{\rho_i} \nabla W_{ij} \omega_j. \quad (3.2)$$

In fig. 3.1, we plot the decay in the velocity magnitude with time for different resolutions

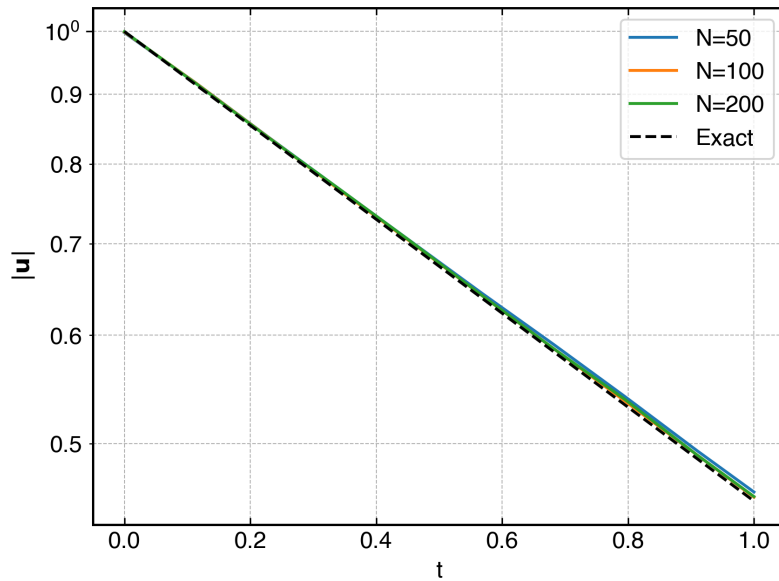


Figure 3.1 : The decay in velocity magnitude for different resolutions compared with the exact solution for the Taylor-Green problem.

compared with the exact solution. Clearly, the decay in the velocity magnitude is very close to the expected result.

In the lid-driven cavity problem, we consider a two-dimensional domain of size $1m \times 1m$ with 5 layers of ghost particles representing the solid particles. The top wall at $y = 1m$ is given a velocity $u = 1m/s$ along the x-direction. We solve the problem

using the L-IPST-C scheme for different resolutions for 10 sec. However, we discretize the viscous term using the method proposed by Cleary et al. (1999) (see section 2.2.3). In

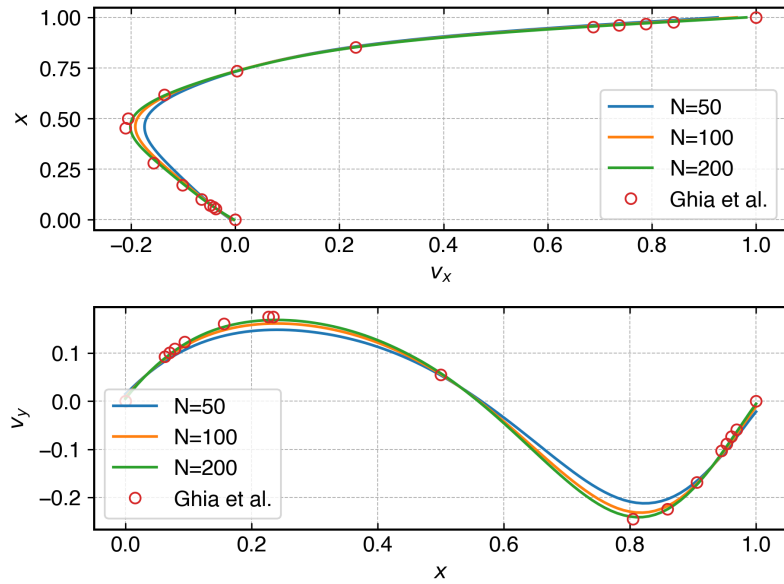


Figure 3.2 : The velocity along x and y directions along the center line $x = 0.5m$ of the domain for the lid-driven cavity problem.

fig. 3.2, we plot the velocity along the centerline $x = 0.5m$ of the domain compared with the result of Ghia et al. (1982). Clearly, the increase in resolution improves the accuracy.

We note that many researchers (as discussed in section 1.5.1) use the above approach to verify their SPH schemes. For both problems discussed above, we use a discretization that is not second-order accurate, as shown in section 2.2. Therefore, these verification techniques are unable to detect issues in specific discretization in a scheme. In addition, the simulations take a significant amount of time. For example, the 200×200 resolution lid-driven cavity case took 150 minutes on an Intel(R) Xeon(R) CPU E5-2650 v3 processor with 40 threads. In the case of the Taylor-Green problem, since the exact solution is known, one can evaluate the L_1 error in velocity or pressure. In fig. 3.3, we plot the L_1 error using eq. (2.42) in velocity as a function of particle spacing. The L_1 error is not second-order and diverges as we increase resolution from 100×100 to 200×200 . Furthermore, this result does not suggest the exact reason for the error.

In general, one cannot exercise specific terms in the governing differential equation in all the methods described above. Therefore, the source of the error cannot be determined. For example, the code may show convergence in the case of the Gresho-Chan vortex problem (an inviscid problem) but fail for the Taylor-Green vortex problem due to an issue with the discretization of the viscous term. It is only recently Antuono (2020) proposed an analytic solution for three-dimensional Navier-Stokes equations has been

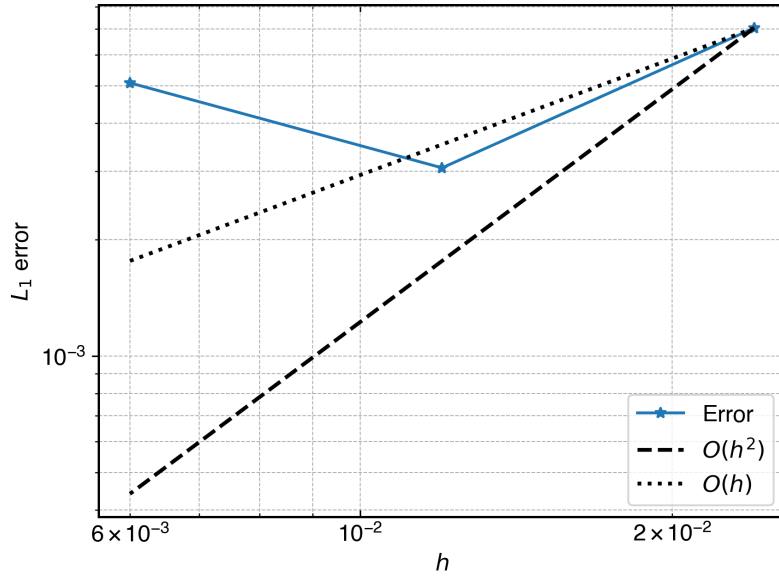


Figure 3.3 : The L_1 error in velocity for the Taylor-Green problem.

proposed. Other recent work of Sharma et al. (2019) has only focused on numerical investigation. Therefore, it is not easy to apply the MES in three dimensions. Furthermore, such studies require an even larger computational effort. Finally, we note that the Taylor-Green vortex problem is for an incompressible fluid making it difficult to test a WCSPH scheme.

Therefore, in the context of SPH, the comparison and MES techniques are insufficient and inefficient. We require a better method to verify the code before proceeding to validation. The method of manufactured solutions offers precisely such a technique, which is described in the next section.

3.2 The method of manufactured solutions

In conventional finite volume and finite element schemes, it is mandatory to demonstrate the order of convergence, and the MMS has been used for this, as discussed in section 1.5. For the SPH method, obtaining second-order convergence has itself been a challenge (Vacondio et al., 2020) as discussed in the previous chapter. Moreover, to the best of our knowledge, the MMS method has not been applied in the context of SPH. In this work, we use the principles of MMS to formally verify SPH codes in a fast and reliable manner. The technique facilitates a careful investigation of the various discretization operators, boundary condition implementation, and time integrators.

In MMS, an *artificial or manufactured solution* is assumed. Let us assume the manufactured solution (MS) for ρ , \mathbf{u} , and p in eq. (2.14) are $\tilde{\rho}$, $\tilde{\mathbf{u}}$, and \tilde{p} , respectively. Since

MS is not the solution of the eq. (2.14), we obtain a residue

$$\begin{aligned} s_\rho &= \frac{d\tilde{\rho}}{dt} + \tilde{\rho}\nabla \cdot \tilde{\mathbf{u}}, \\ \mathbf{s}_u &= \frac{d\tilde{\mathbf{u}}}{dt} + \frac{\nabla\tilde{p}}{\tilde{\rho}} - \nu\nabla^2\tilde{\mathbf{u}}, \end{aligned} \quad (3.3)$$

where s_ρ and \mathbf{s}_u are the residue term for continuity and momentum equation, respectively. Since we have the closed-form expression for all the terms in the RHS of eq. (3.3), we may introduce the residue terms as source terms in the governing equations. We write the modified governing equations as

$$\begin{aligned} \frac{d\rho}{dt} &= -\rho\nabla \cdot \mathbf{u} + s_\rho, \\ \frac{d\mathbf{u}}{dt} &= -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{u} + \mathbf{s}_u. \end{aligned} \quad (3.4)$$

Finally, we solve eq. (3.4). The addition of the source terms ensures that the solution is $\tilde{\rho}$, $\tilde{\mathbf{u}}$, and \tilde{p} .

One must take a few precautions while constructing an artificial solution in the MMS (Salari et al., 2000):

1. The MS must be C^n smooth where n is the order of the governing equations.
2. It must exercise all the terms, i.e., for any evolution equation, all the MS cannot be time-independent.
3. The MS must be bounded in the domain of interest. For example, the MS $u = \tan(x)$ in the domain $[-\pi, \pi]$ is not bounded thus, should not be used.
4. The MS should not prevent the successful completion of the code. For example, if the code assumes the solution to have positive pressure, then the MS must make sure that the pressure is not negative.
5. The MS should make sure that the solution satisfies the basic physics. For example, the flux must be continuous in a shear layer flow with discontinuous viscosity.

We note that the MS may not be physically realistic. We modify the basic steps for MMS proposed by Oberkampf et al. (2010) for use in the context of WCSPH as follows:

1. Obtain the modified form of the governing equations as employed in the scheme. For example, in the case of the δ -SPH scheme (Antuono et al., 2010), the continuity equation used is

$$\frac{d\rho}{dt} = -\rho\nabla \cdot \mathbf{u} + D\nabla^2\rho, \quad (3.5)$$

where $D = \delta hc_o$ is the damping used, and δ is a numerical parameter. The additional diffusive term in eq. (3.5) must be retained while obtaining the source term.

2. Construct the MS using analytical functions. The general form of MS is given by

$$f(x, y, t) = \phi_o + \phi(x, y, t), \quad (3.6)$$

where f is any property viz. ρ , \mathbf{u} , or p ; ϕ_o is a constant, and $\phi(x, y, t)$ is a function chosen such that the five precautions listed above are satisfied.

3. Obtain the residue due to the manufactured solution as done in eq. (3.3).
4. Add the residue as a source term in the code appropriately. In SPH, the source term $s = s(x, y, z, t)$, is discretized as $s_i = s(x_i, y_i, z_i, t)$ where subscript i denotes the i^{th} particle.
5. Solve the modified equations using the code for different particle spacings/smoothing lengths. The properties of the boundary particles are updated using the MS. We note that in the context of WCSPH schemes, one should not evaluate the derived quantities like the gradient of the velocity field using the MS on the solid boundary.
6. Evaluate the discretization error for each resolution. We evaluate the error using eq. (2.42).
7. Compute the order of accuracy and determine whether the desired order is achieved.

The code involves discretization of the governing equations and appropriate implementation of the boundary conditions. The MMS can be used to determine the accuracy of both. However, to obtain the accuracy of boundary condition implementations, the order of convergence of the governing equations should be at least the order of the boundary condition implementations (Choudhary et al., 2016).

3.2.1 Implementation of MMS

In this section, we discuss the implementation of MMS for a general meshless method code. In order to avoid mistakes while evaluating the source term. We do not perform the differentiation manually. In this work, we use `sympy` (Meurer et al., 2017) tool along with the `mako` python packages. We evaluate the gradient of a scalar as shown in listing 3.1.

Listing 3.1 : Code for gradient of a scalar.

```
import sympy as sp
def grad(pres):
    return [sp.diff(pres, x),
```

```

    sp.diff(pres, y),
    sp.diff(pres, z)]

```

Similarly, the gradient of a vector is evaluated as shown in listing 3.2.

Listing 3.2 : Code for gradient of a vector.

```

def vec_grad(vel):
    return [
        sp.diff(vel[0], x),
        sp.diff(vel[0], y),
        sp.diff(vel[0], z),
        sp.diff(vel[1], x),
        sp.diff(vel[1], y),
        sp.diff(vel[1], z),
        sp.diff(vel[2], x),
        sp.diff(vel[2], y),
        sp.diff(vel[2], z)]

```

We evaluate the divergence of a vector as shown in listing 3.3.

Listing 3.3 : Code for divergence of a vector.

```

def div(vel):
    return (sp.diff(vel[0], x) +
            sp.diff(vel[1], y) +
            sp.diff(vel[2], z))

```

The Laplacian is evaluated using the gradient and divergence functions as shown in listing 3.4.

Listing 3.4 : Code for Laplacian of a vector.

```

def laplace(vel):
    gradv = vec_grad(vel)
    return [div(gradv[0:3]), div(gradv[3:6]), div(gradv[6:9])]

```

Using the functions for gradient, divergence, and Laplacian evaluation, we can obtain the source term for any given MS $\tilde{\rho}$, \tilde{p} , and $\tilde{\mathbf{u}}$ as shown in listing 3.5.

Listing 3.5 : Code for evaluation of source term from continuity and momentum equations.

```

def continuity(rhoc, vel, rhocs, delta_coeff=0.0):
    srho = sp.diff(rhoc, t) + dot(grad(rhoc), [us, vs, ws]) + \
            rhocs * div(vel) - delta_coeff*h * laplace_scal(rhoc)

```

```

return srho

def momentum_eq(vel, rhoc, p, rhocs, nu=0.0, comp=0):
    adv = vec_mat_mul(vec_grad(vel), [us, vs, ws])
    s_u = sp.diff(vel[comp], t) + adv[comp] + \
        grad(p)[comp]/rhocs - nu*laplace(vel)[comp]
    return s_u

```

In listing 3.5, we use **rhoc**, **p**, **vel**, **rhocs**, and **nu** for ρ , p , \mathbf{u} , ρ and ν , respectively. The **comp** parameter is used to pass the component vector output. On evaluating the expression for the source term, we save the expression in YAML file format. An example of the format is given in listing 3.6.

Listing 3.6 : YAML format for saving the source terms.

```

py:
  mms1:
    gradv0: 2*pi*cos(2*pi*x)*cos(2*pi*y)
    gradv1: -2*pi*sin(2*pi*x)*sin(2*pi*y)
    gradv2: '0'
    gradv3: 2*pi*sin(2*pi*x)*sin(2*pi*y)
    gradv4: -2*pi*cos(2*pi*x)*cos(2*pi*y)
    gradv5: '0'
    gradv6: '0'
    gradv7: '0'
    gradv8: '0'
    p: cos(4*pi*x) + cos(4*pi*y)
    rhoc: rhoc0 + (cos(4*pi*x) + cos(4*pi*y))/c0**2
    spp: -4*pi*u*sin(4*pi*x) - 4*pi*v*sin(4*pi*y)
    srho: -4*pi*u*sin(4*pi*x)/c0**2 - 4*pi*v*sin(4*pi*y)/c0**2
    su: 2*pi*u*cos(2*pi*x)*cos(2*pi*y) - \
        2*pi*v*sin(2*pi*x)*sin(2*pi*y) - 4*pi*sin(4*pi*x)/rhoc
    sv: 2*pi*u*sin(2*pi*x)*sin(2*pi*y) - \
        2*pi*v*cos(2*pi*x)*cos(2*pi*y) - 4*pi*sin(4*pi*y)/rhoc
    sw: '0.0'
    u: sin(2*pi*x)*cos(2*pi*y)
    v: -sin(2*pi*y)*cos(2*pi*x)
    w: '0.0'

```

We use the YAML file saved in this format to generate the python code in PySPH framework using mako template as shown in listing 3.7.

Listing 3.7 : Mako template for generating the source code

```
def get_props(x, y, z, t, c0):
    from numpy import sin, cos, exp, log
    u = (${formula['u']}) * np.ones_like(x)
    v = (${formula['v']}) * np.ones_like(x)
    w = (${formula['w']}) * np.ones_like(x)
    p = (${formula['p']}) * np.ones_like(x)
    rhoc = p/c0**2 + 1.0

    return u, v, w, rhoc, p

class AddMomentumSourceTerm(Equation):
    def initialize(self, d_au, d_av, d_aw, d_idx):
        d_au[d_idx] = 0.0
        d_av[d_idx] = 0.0
        d_aw[d_idx] = 0.0

    def post_loop(self, d_au, d_av, d_aw, d_idx, d_x, d_y, d_z,
                  d_u, d_v, d_w, d_rho, d_rhoc, d_p, t, dt, d_c0):

        x = d_x[d_idx]
        y = d_y[d_idx]
        z = d_z[d_idx]
        c0 = d_c0[0]
        rhoc0 = 1.0
        rho0 = 1.0

        u = ${formula['u']}
        v = ${formula['v']}
        w = ${formula['w']}
        p = ${formula['p']}
        rhoc = p/c0**2 + rhoc0
        rho = d_rho[d_idx]

        d_au[d_idx] += ${formula['su']}
```

```

d_av[d_idx] += ${formula['sv']}
d_aw[d_idx] += ${formula['sw']}

```

A simple code to generate the python code from the YAML file and mako template is given in listing 3.8.

Listing 3.8 : Python code to generate code from mako template file.

```

mytemplate = Template(filename=mako_file)
yaml_file = os.path.join(dirname, 'mms.yaml')
data = yaml.load(fp, Loader=yaml.FullLoader)['py']
for key in data:
    code = mytemplate.render(formula=data[key])
    dirname = os.path.dirname(__file__)
    outfile = os.path.join(dirname, key+'.py')
    fileptr = open(outfile, 'w')
    fileptr.writelines(code)
    fileptr.close()

```

Therefore, using the above implementation, we only need to pass the manufactured solution using a python expression, and it automatically generates a PySPH Equation subclass for the MS. In this process, the chances of human error are very low, and we can test the convergence of our code with high confidence.

In the next section, we demonstrate the application of MMS to obtain the order of convergence for the schemes discussed in the previous chapter.

3.3 Application of the MMS

In this section, we apply the MMS to obtain the order of convergence of various schemes. We first determine the initial particle configuration viz. unperturbed, perturbed, or packed required for the MMS. We then demonstrate that one can apply the MMS to arbitrarily-shaped domains. We then compare the EDAC and PE-IPST-C schemes which differ in the treatment of the density. We next apply the MMS to E-C and TV-C schemes as they employ different governing equations compared to standard WCSPH in eq. (2.14). We also demonstrate the application of the MMS method as a technique to identify mistakes in the implementation. We then show the application of the MMS to obtain convergence in the 3D domain and extreme resolutions.

In all our test cases, we use the quintic spline kernel with $h_{\Delta x} = 1.2$, where Δx is the initial inter-particle spacing. We consider a domain of size $1m \times 1m$. We simulate all the test cases for 50×50 , 100×100 , 200×200 , 250×250 , 400×400 , 500×500 , and 1000×1000

resolutions (unless stated otherwise) to obtain the order of convergence plots. In all our simulations, we initialize the particle properties using the MS. We then solve eq. (3.4) and set the properties on any solid particle using the MS before every timestep. We set a fixed time step corresponding to the highest resolution for all the other resolutions as discussed in section 2.4.1. We evaluate the L_1 error using eq. (2.42) in the solution.

3.3.1 The effect of initial particle configuration

The initial particle configuration plays a significant role in the error estimation since the divergence of the velocity is captured accurately when the particles are uniformly arranged (see section 2.2.1). In this test case, we consider three different initial configurations of particles, widely used in SPH literature, viz. unperturbed, perturbed, and packed. The unperturbed configuration is where we place the particles on a Cartesian grid such that the particles are at a constant distance along the grid lines. In the perturbed configuration, we perturb the particles initially placed on a Cartesian grid by adding a uniformly distributed random displacement as a fraction of the inter-particle spacing Δx . For the packed configuration, we use the method proposed in Colagrossi et al. (2012) to resettle the particles from a randomly perturbed distribution to a new configuration such that the particle density ψ of the particles is nearly constant. In fig. 3.4, we show all the initial particle distributions with the solid boundary particles in orange.

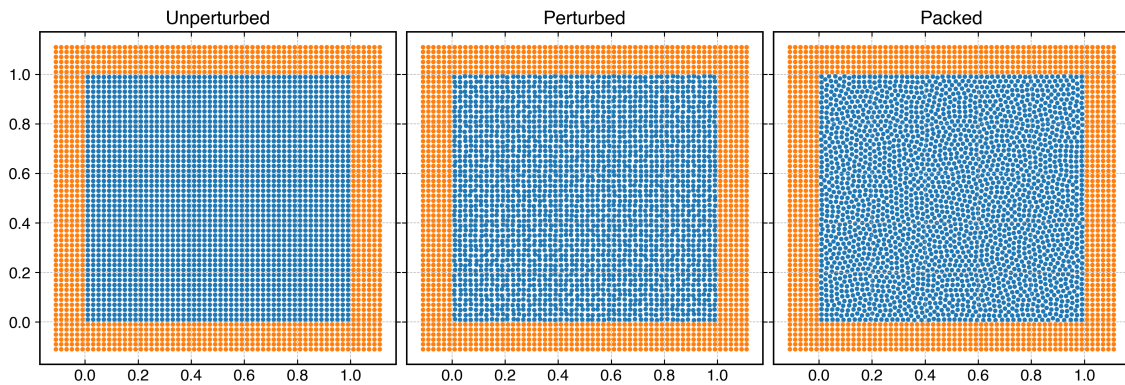


Figure 3.4 : The different initial particle arrangements in blue with the solid boundary in orange.

We consider the MS of the form

$$\begin{aligned}
 u(x, y, t) &= e^{-10t} \sin(2\pi x) \cos(2\pi y), \\
 v(x, y, t) &= -e^{-10t} \sin(2\pi y) \cos(2\pi x), \\
 p(x, y, t) &= e^{-10t} (\cos(4\pi x) + \cos(4\pi y)), \\
 \rho(x, y, t) &= \frac{p}{c_o^2} + \rho_o,
 \end{aligned} \tag{3.7}$$

where, we set $c_o = 20m/s$ for all our test cases. The MS complies with all the required conditions discussed in section 3.2. We note that the MS chosen resembles the exact solution of the Taylor-Green problem. However, since the code simulates the NS equation using a weakly compressible formulation, we obtain additional source terms when we substitute the MS to eq. (2.14) with $\nu = 0.01m^2/s$. We obtain the source terms from the symbolic framework, `sympy` as

$$\begin{aligned}
s_u(x, y, t) &= 2\pi u e^{-10t} \cos(2\pi x) \cos(2\pi y) - 2\pi v e^{-10t} \sin(2\pi x) \sin(2\pi y) - \\
&\quad 10e^{-10t} \sin(2\pi x) \cos(2\pi y) + 0.08\pi^2 e^{-10t} \sin(2\pi x) \cos(2\pi y) - \\
&\quad \frac{4\pi e^{-10t} \sin(4\pi x)}{\rho}, \\
s_v(x, y, t) &= 2\pi u e^{-10t} \sin(2\pi x) \sin(2\pi y) - 2\pi v e^{-10t} \cos(2\pi x) \cos(2\pi y) - \\
&\quad 0.08\pi^2 e^{-10t} \sin(2\pi y) \cos(2\pi x) + 10e^{-10t} \sin(2\pi y) \cos(2\pi x) - \\
&\quad \frac{4\pi e^{-10t} \sin(4\pi y)}{\rho}, \\
s_\rho(x, y, t) &= -\frac{4\pi u e^{-10t} \sin(4\pi x)}{c_0^2} - \frac{4\pi v e^{-10t} \sin(4\pi y)}{c_0^2} - \frac{10(\cos(4\pi x) + \cos(4\pi y)) e^{-10t}}{c_0^2}.
\end{aligned} \tag{3.8}$$

We add $\mathbf{s}_u = s_u \hat{\mathbf{i}} + s_v \hat{\mathbf{j}}$ to the momentum equation and s_ρ to the continuity equation as shown in eq. (3.4). We solve the modified WCSPH equations in eq. (3.4) using the L-IPST-C method for 100 timesteps where we initialize the domain using the MS in eq. (3.7) at $t = 0$. The values of the properties \mathbf{u} , p , and ρ on the (orange) solid particles are set using the MS in eq. (3.7) at the start of every time step.

In fig. 3.5, we plot the L_1 error in pressure and velocity after 10 timesteps as a function of resolution for different initial particle distributions. Clearly, the difference in initial configuration affects the error in pressure by a large amount. However, in velocity, the error is large in the case of the perturbed configuration only. The unperturbed configuration has zero divergence error at $t = 0$ (see section 2.3.2). Whereas the perturbed configuration has a high error due to the random initialization. Over the course of a few iterations, there is no significant difference between the distribution of particles for the unperturbed and the packed configurations. Therefore, we simulate the problems for 100 timesteps for a fair comparison.

In fig. 3.6, we plot the L_1 error in pressure and velocity after 100 timesteps as a function of resolution for the cases considered. Clearly, the difference in error is reduced. However, the order of convergence is not captured accurately. This is because the initial divergence is not captured accurately by the packed and perturbed configurations as dis-

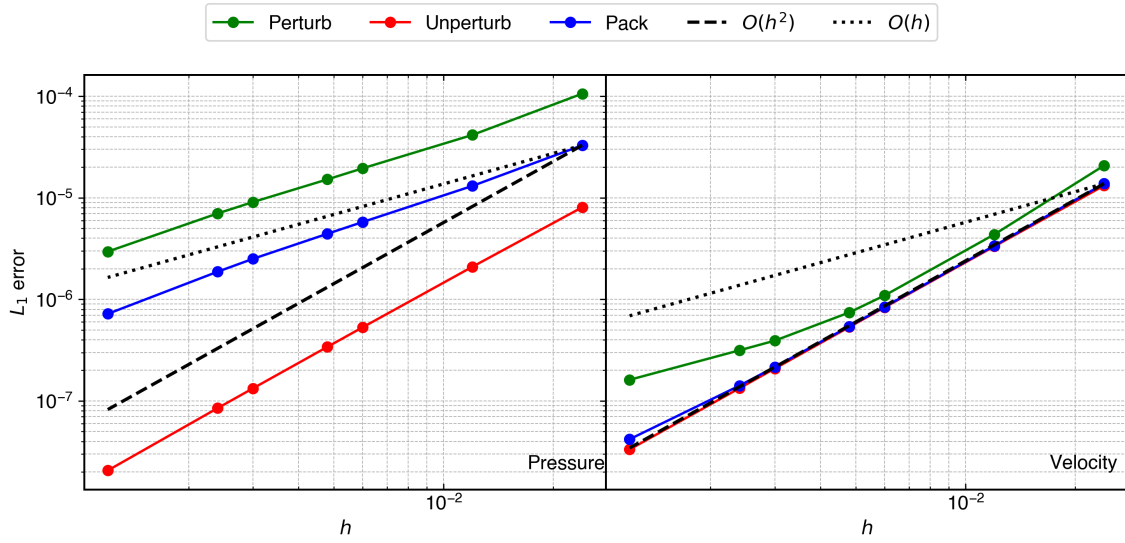


Figure 3.5 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.7) and the source term in eq. (3.8) after 10 timesteps for different configurations.

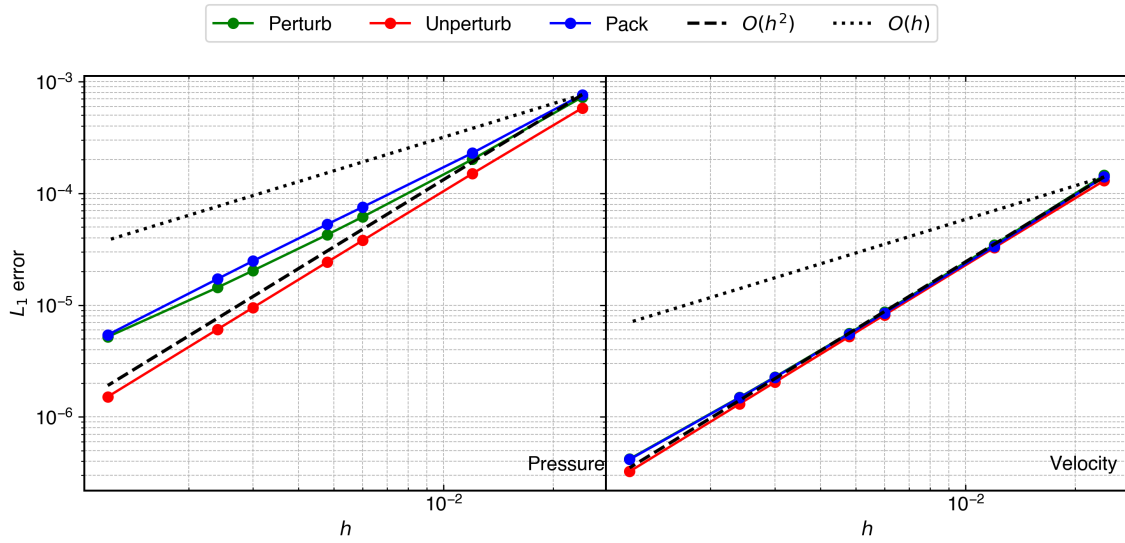


Figure 3.6 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.7) and the source term in eq. (3.8) after 100 timesteps for all the configurations.

cussed in section 2.3.2. This difference can be avoided through the use of a non-solenoidal velocity field. Therefore we consider the following modified MS, given by

$$\begin{aligned}
 u(x, y, t) &= y^2 e^{-10t} \sin(2\pi x) \cos(2\pi y), \\
 v(x, y, t) &= -e^{-10t} \sin(2\pi y) \cos(2\pi x), \\
 p(x, y, t) &= (\cos(4\pi x) + \cos(4\pi y)) e^{-10t}.
 \end{aligned} \tag{3.9}$$

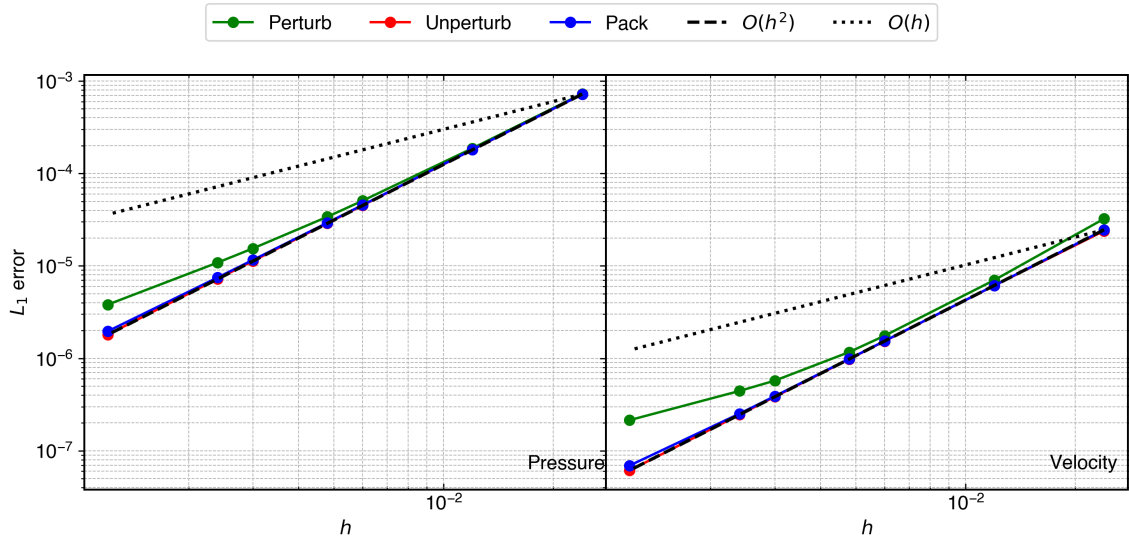


Figure 3.7 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.9) and the corresponding source terms after 100 timesteps for all the configurations.

We note that the new MS velocity field is not divergence-free. We obtain the source term with $\nu = 0.01m^2/s$ as done in eq. (3.8). We simulate the problem by initializing the domain using MS in eq. (3.9). We also update the solid boundary properties using this MS before every timestep. In fig. 3.7, we plot the L_1 error for pressure and velocity as a function of resolution. Clearly, both the packed and unperturbed domains show second-order convergence. Whereas the perturbed configuration fails to show second-order convergence. Therefore, in the context of WCSPH schemes, one should not use a divergence-free field in the MS. Furthermore, one should use either a packed or unperturbed configuration for the convergence study.

It is important to note that in stark contrast to the Taylor-Green vortex problem, the method shows second-order convergence irrespective of the value of c_o . In the previous chapter, a much higher $c_o = 80m/s$ was necessary to demonstrate second-order convergence. Furthermore, the convergence is independent of the initial configuration after 100 steps; therefore, we recommend simulating all the test cases for at least 100 timesteps to obtain the actual order of convergence. It is important to note that some discretizations are second-order accurate only when an unperturbed configuration is used (see section 2.2). Therefore, to test the robustness of the discretization, we recommend using a packed configuration.

3.3.2 The selection of the domain shape

We now show the effect of the shape of the domain on the convergence of a scheme. We consider a square-shaped and a butterfly-shaped domain as shown in fig. 3.8. We consider the MS with the non-solenoidal velocity field in eq. (3.9) as used in the previous test case. The source terms obtained remain the same as before, where we consider $\nu = 0.01\text{m}^2/\text{s}$. We solve the modified equations using the L-IPST-C scheme for 100 timesteps for each domain. We initialize the fluid and solid particles using the MS in eq. (3.9). We update the properties of the solid particles before every timestep using the same MS.

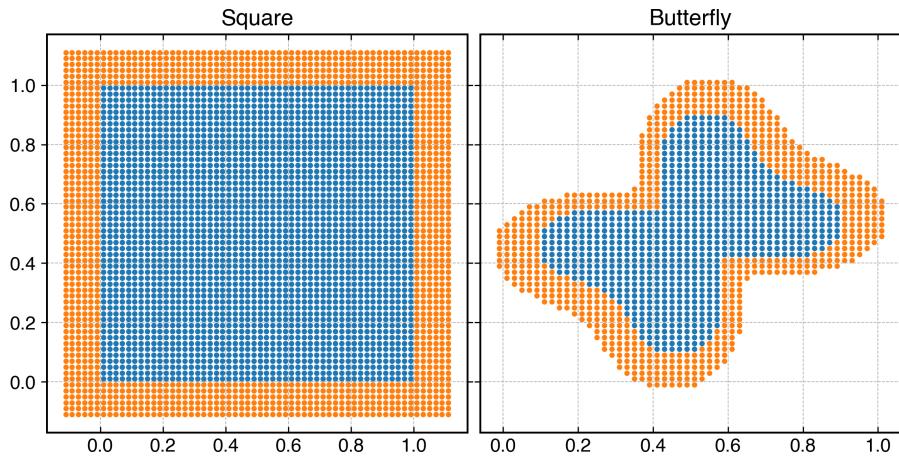


Figure 3.8 : The different domain shapes with solid particles in orange and fluid particles in blue.

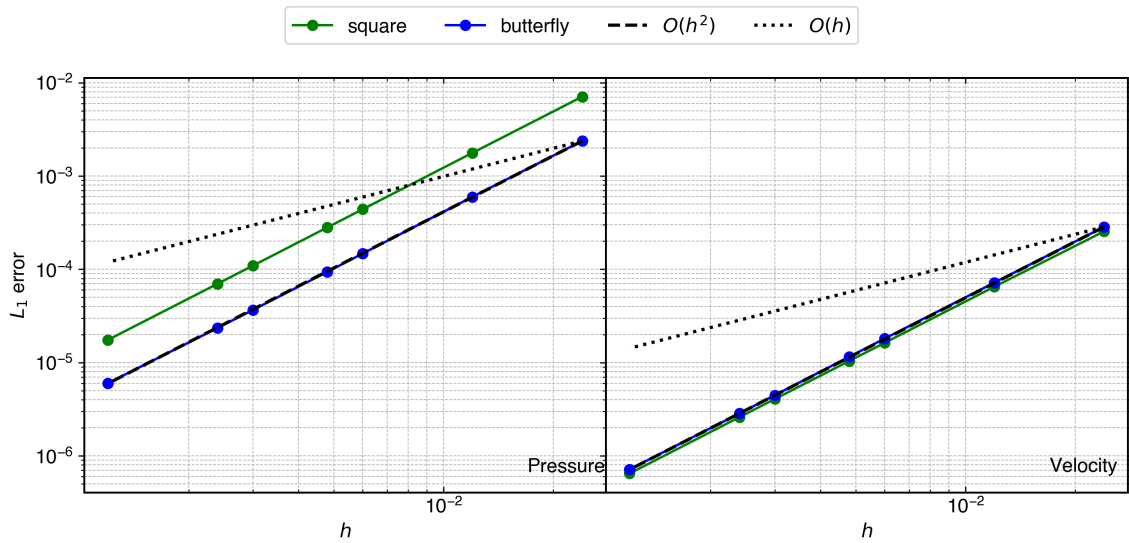


Figure 3.9 : The L_1 error in pressure (left) and velocity (right) with increase in resolution for different shapes of the domain.

In fig. 3.9, we show the L_1 error after 100 timesteps in pressure and velocity as a function of resolution for both domains considered. Clearly, both domains considered show second-order convergence. Hence, one can consider any shape of the domain for the convergence study of WCSPH schemes using MMS. However, we only use a square-shaped domain for all our test cases.

3.3.3 Comparison of EDAC and PE-IPST-C

In this test case, we compare the convergence of EDAC (see appendix A.5.4) and PE-IPST-C (see section 2.3.5) schemes. These two schemes have two major differences. First, the discretizations used in the PE-IPST-C method are all second-order accurate in contrast to the EDAC scheme. Second, in PE-IPST-C, the density ρ is a transport property. We evaluate ρ by inverting the linear equation of state given in eq. (2.32).

In the EDAC scheme, the initial configuration of particles affects the results. Therefore, we consider an unperturbed configuration, as shown in fig. 3.4. In order to reduce the complexity, we consider an inviscid MS ($\nu = 0$) given by

$$\begin{aligned} u(x, y) &= \sin(2\pi x) \cos(2\pi y), \\ v(x, y) &= -\sin(2\pi y) \cos(2\pi x), \\ p(x, y) &= \cos(4\pi x) + \cos(4\pi y). \end{aligned} \tag{3.10}$$

Thus, the code must maintain the pressure and velocity fields in the absence of the viscosity. The source term for the EDAC scheme is given by

$$\begin{aligned} s_u(x, y) &= 2\pi u \cos(2\pi x) \cos(2\pi y) - 2\pi v \sin(2\pi x) \sin(2\pi y) - \frac{4\pi \sin(4\pi x)}{\psi}, \\ s_v(x, y) &= 2\pi u \sin(2\pi x) \sin(2\pi y) - 2\pi v \cos(2\pi x) \cos(2\pi y) - \frac{4\pi \sin(4\pi y)}{\psi}, \\ s_p(x, y) &= -1.25h \left(-16\pi^2 \cos(4\pi x) - 16\pi^2 \cos(4\pi y) \right) - 4\pi u \sin(4\pi x) - 4\pi v \sin(4\pi y). \end{aligned} \tag{3.11}$$

We note that the source term employs density $\rho = \psi$ which is the function of particle position. In the case of the PE-IPST-C scheme, the source term is given by

$$\begin{aligned} s_u(x, y) &= 2\pi u \cos(2\pi x) \cos(2\pi y) - 2\pi v \sin(2\pi x) \sin(2\pi y) - \frac{4\pi \sin(4\pi x)}{\rho}, \\ s_v(x, y) &= 2\pi u \sin(2\pi x) \sin(2\pi y) - 2\pi v \cos(2\pi x) \cos(2\pi y) - \frac{4\pi \sin(4\pi y)}{\rho}, \\ s_p(x, y) &= -1.25h \left(-16\pi^2 \cos(4\pi x) - 16\pi^2 \cos(4\pi y) \right) - 4\pi u \sin(4\pi x) - 4\pi v \sin(4\pi y). \end{aligned} \tag{3.12}$$

We note that the source term s_p in eq. (3.11) and eq. (3.12) are same. We simulate the problem with the MS in eq. (3.10). The solid (shown in orange) boundary properties are reset using this MS before every time step.

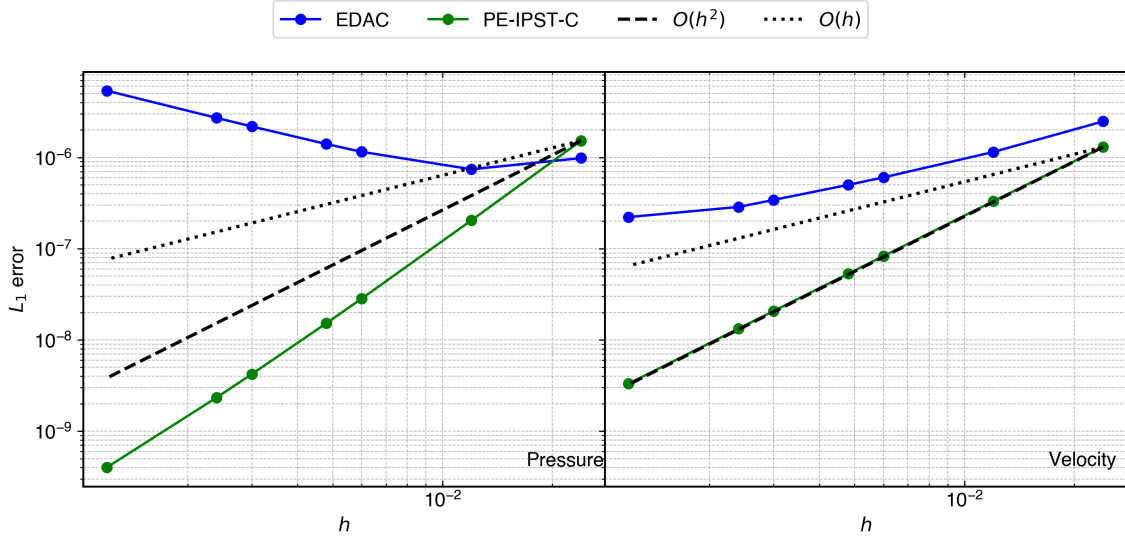


Figure 3.10 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.10), and the source term in eq. (3.11) for EDAC and eq. (3.12) for PE-IPST-C after one timestep.

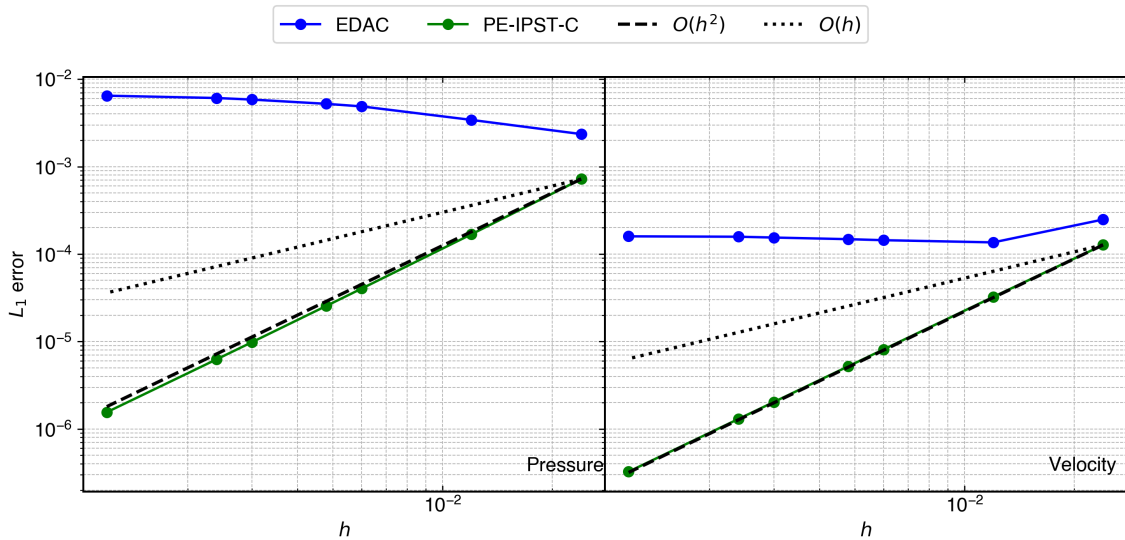


Figure 3.11 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.10), and the source term in eq. (3.11) for EDAC and eq. (3.12) for PE-IPST-C after 100 timesteps.

In fig. 3.10, we plot the L_1 error in pressure and velocity after one timestep for both schemes. Clearly, the EDAC case diverges in the case of pressure, whereas we observe a reduced order of convergence in velocity. In contrast, the PE-IPST-C scheme shows

second-order convergence in velocity and higher in the case of pressure. We observe this increased order only for the first iteration. In fig. 3.11, we plot the L_1 error in pressure and velocity after 100 timesteps for both schemes. In the case of the EDAC scheme, the order of convergence in the velocity does not remain first-order, whereas the L-IPST-C scheme shows second-order convergence in both pressure and velocity.

We note that we use an unperturbed mesh; therefore, we must obtain second-order convergence up to the order of error due to numerical quadrature (see eq. (1.17)) for one timestep in the case of the EDAC scheme as well. However, we observe a divergence in pressure. This behavior occurs since $\rho = \psi$ (a function of neighbor particle positions) is present in the source term in eq. (3.11). However, the density in the source term expression should be a fluid property as it is present in the governing differential equation. Therefore, as mentioned in the previous chapter, we should treat density in the governing equation as a transport property as we do in the case of the PE-IPST-C scheme.

3.3.4 Comparison of E-C and TV-C

In this test case, we apply MMS to E-C and TV-C schemes introduced in section 2.3.5. The governing equations for the E-C scheme are given in eq. (2.40) whereas for TV-C in eq. (2.36). The expression for the source terms turns out to be the same for both eq. (2.40) and eq. (2.36) governing equations given by

$$\begin{aligned} s_\rho &= \frac{\partial \rho}{\partial t} + \rho \nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla \rho, \\ s_{\mathbf{u}} &= \frac{\partial \mathbf{u}}{\partial t} + \frac{\nabla p}{\rho} - \nu \nabla^2 \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}. \end{aligned} \quad (3.13)$$

These source terms are the same as obtained in the case of the L-IPST-C scheme as well. In the E-C scheme, we fix the grid and add the convective term as the correction, whereas in the TV-C scheme, we add the shifting velocity in the LHS of the governing equations.

In order to show the convergence of the scheme, we consider the inviscid MS in eq. (3.10) with the linear EOS. We do not consider the viscous term since the term introduces similar errors in both schemes. We write the source term as

$$\begin{aligned} s_u(x, y) &= 2\pi u \cos(2\pi x) \cos(2\pi y) - 2\pi v \sin(2\pi x) \sin(2\pi y) - \frac{4\pi \sin(4\pi x)}{\rho}, \\ s_v(x, y) &= 2\pi u \sin(2\pi x) \sin(2\pi y) - 2\pi v \cos(2\pi x) \cos(2\pi y) - \frac{4\pi \sin(4\pi y)}{\rho}, \\ s_\rho(x, y) &= -\frac{4\pi u \sin(4\pi x)}{c_0^2} - \frac{4\pi v \sin(4\pi y)}{c_0^2}, \end{aligned} \quad (3.14)$$

where $\mathbf{s}_{\mathbf{u}} = s_u \hat{\mathbf{i}} + s_v \hat{\mathbf{j}}$ is the source term for the momentum equation in both schemes. We consider an unperturbed initial particle distribution and run the simulation for 100

timesteps. The particles are initialized with the MS in eq. (3.10) and solid boundary are reset using the MS before every time step.

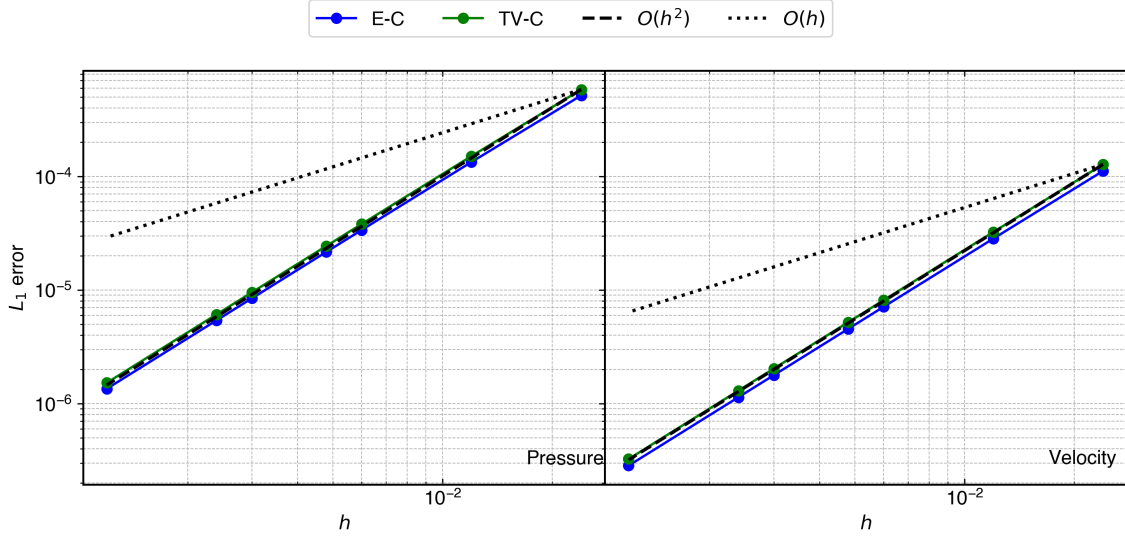


Figure 3.12 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.10) and the source term in eq. (3.14) after 100 timesteps for the different schemes.

In fig. 3.12, we plot the L_1 error in pressure and velocity as a function of resolution for both schemes. Since we use second-order accurate discretization in both schemes, they show second-order convergence in both pressure and velocity, as expected. Thus, we see that the modified governing equations (eq. (2.36) and eq. (2.40)) must be considered to obtain the source term for the schemes.

3.3.5 Identification of mistakes in the implementation

In this section, we demonstrate the use of the MMS as a technique to identify mistakes in the implementation. We use the L-IPST-C scheme and introduce either erroneous or lower-order discretization for a specific term in the governing equations. We then use the proposed MMS to identify the problem.

Wrong divergence estimation

We introduce an error in the discretized form of the continuity equation used in the L-IPST-C scheme. We refer to this modified scheme as *incorrect CE*. We write the *incorrect* discretization for the divergence of velocity as

$$\langle \nabla \cdot \mathbf{u} \rangle = \sum_j (\mathbf{u}_j + \mathbf{u}_i) \cdot B_i \nabla W_{ij} \omega_j, \quad (3.15)$$

where the error is shown in red. Since only the continuity equation is involved, we use the inviscid MS given by

$$\begin{aligned} u(x, y) &= (y - 1)^2 \sin(2\pi x) \cos(2\pi y), \\ v(x, y) &= -\sin(2\pi y) \cos(2\pi x), \\ p(x, y) &= (y - 1)(\cos(4\pi x) + \cos(4\pi y)). \end{aligned} \quad (3.16)$$

The source terms can be determined by subjecting the above MS to eq. (2.14). We simulate the problem for one timestep with a packed domain (see fig. 3.4). In order to test erroneous or lower-order discretization in the scheme, we recommend the simulation to only one timestep with a packed initial particle distribution.

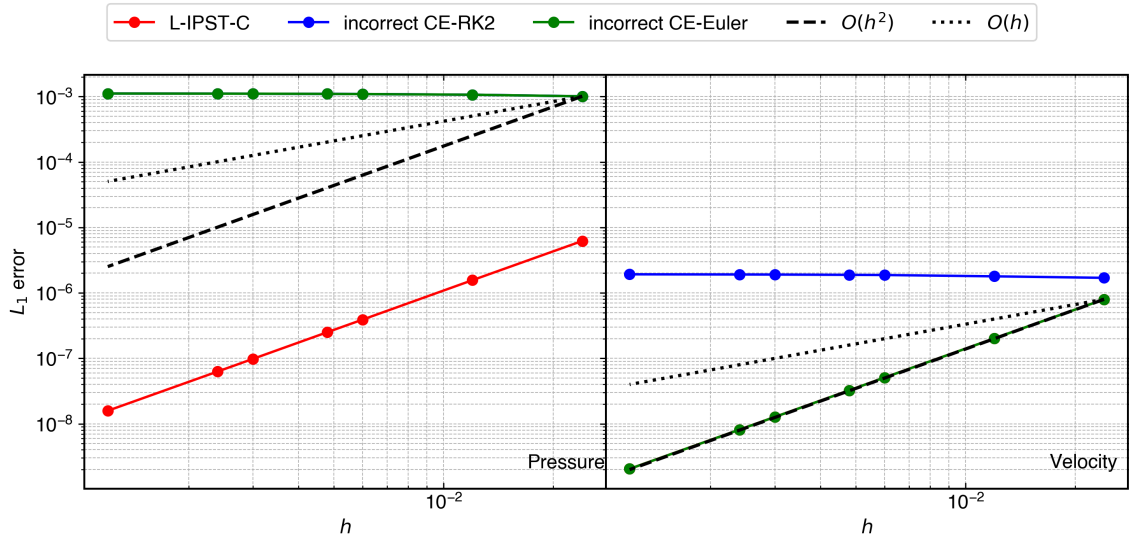


Figure 3.13 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.16) and the corresponding source term after 1 timestep for L-IPST-C and the scheme where the divergence is computed using the incorrect eq. (3.15).

In fig. 3.13, we plot the L_1 error in pressure and velocity as a function of the resolution for the L-IPST-C scheme and its variant *incorrect CE* with two time-integrators, Euler and Runge-Kutta 2nd order (RK2). Clearly, the error in pressure increases by a significant amount, and the order of convergence is zero for *incorrect CE*. However, the error in pressure propagates to velocity in the case of the RK2 integrator, as it is a two-stage integrator. Therefore, we recommend that one must use single-stage integrators while using MMS as a technique to identify mistakes. By looking at the *incorrect CE-Euler* plot in fig. 3.13, we can immediately infer that there is an error in either the continuity equation or the equation of state.

Using a symmetric pressure gradient discretization

In this test case, we use a symmetric formulation as used by Monaghan (2005) and Sun et al. (2017) for the pressure gradient term in the L-IPST-C scheme. We refer to this method as *sym*. Since only the pressure gradient is involved, we use the same MS, as in the previous case.

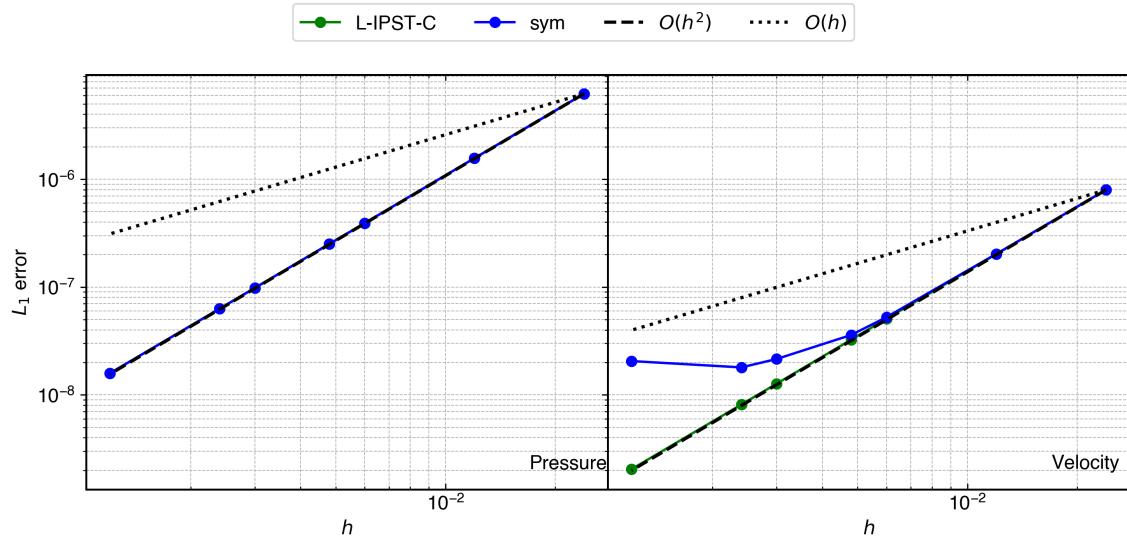


Figure 3.14 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.16) and the corresponding source term after one timestep for L-IPST-C and the scheme where the pressure gradient is computed using symmetric formulation.

In fig. 3.14, we plot the L_1 error after one timestep in pressure and velocity as a function of resolution for L-IPST-C and *sym* schemes. Clearly, the order of convergence is affected in the case of the velocity only. Therefore, it is evident that an inconsistent pressure gradient discretization is used.

Using inconsistent discrete viscous operator

In this test case, we use the formulation proposed by Cleary et al. (1999) to approximate the viscous term in the L-IPST-C scheme. We refer to this method as *Cleary*. Since viscosity is involved, we use the MS involving viscous effect given by eq. (3.9). While testing the viscous term, we use a high value of $\nu = .25m^2/s$ such that the error due to viscosity dominates the error in the momentum equation. We simulate the problem with a packed configuration of particles for one timestep using the MS in eq. (3.9) and with the corresponding source terms.

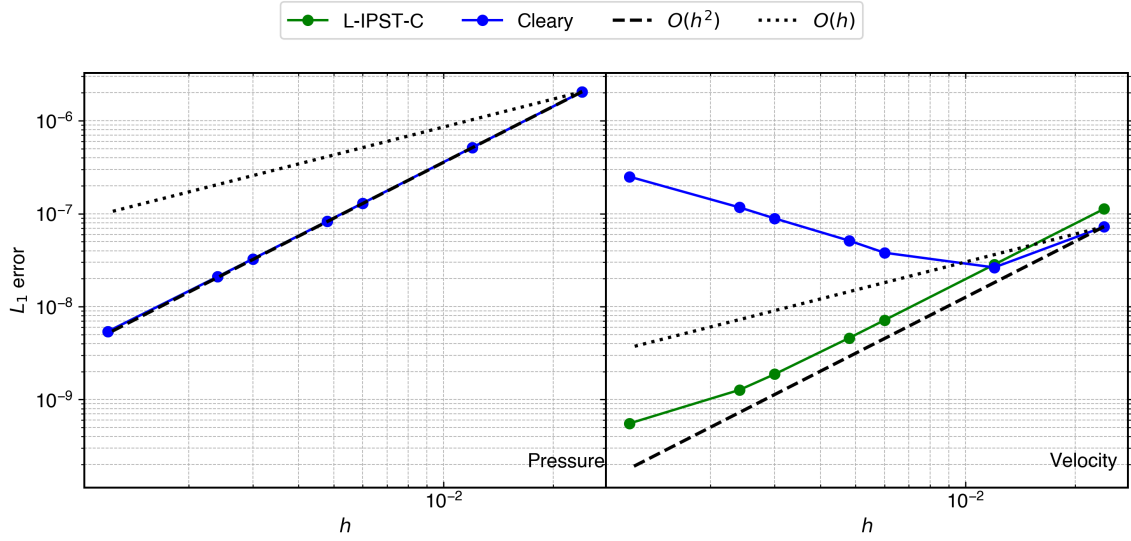


Figure 3.15 : The error in pressure (left) and velocity (right) with fluid particles initialized using the MS in eq. (3.9) and the corresponding source term after one timestep for L-IPST-C and the scheme with the viscous term discretized using formulation given by Cleary et al. (1999).

In fig. 3.15, we plot the L_1 error in pressure and velocity as a function of resolution for L-IPST-C and *Cleary* schemes. Since the viscous formulation by Cleary et al. (1999) shows a negative rate of convergence with the increase in resolution in the perturbed domain (see section 2.2.3), we observe divergence in the velocity. Therefore, we infer that there is an error in the discretization of the viscous term.

3.3.6 Convergence at extreme resolutions

Thus far, we have used particle resolutions in the range $10^{-3} \leq \Delta x \leq 2 \times 10^{-2}$. We wish to study the convergence of the scheme when much higher resolutions are considered. We consider a domain of size 1×1 with uniformly distributed particles as shown in fig. 3.16. In order to reduce computation, we reduce the size of the domain by half if the number of particles crosses $1M$. In the fig. 3.16, the red box shows the domain considered for the computation with one million particles with $\Delta x = 1.25 \times 10^{-4}$. In order to obtain an unbiased error estimate, we consider the same MS and the domain shown by the black box in fig. 3.16 to evaluate

$$L_\infty = \max\{|f_i(x, y, z) - f_o(x, y, z)| \forall i \in N\}, \quad (3.17)$$

where f is the property of interest and f_o is the value of the property evaluated using the MS.

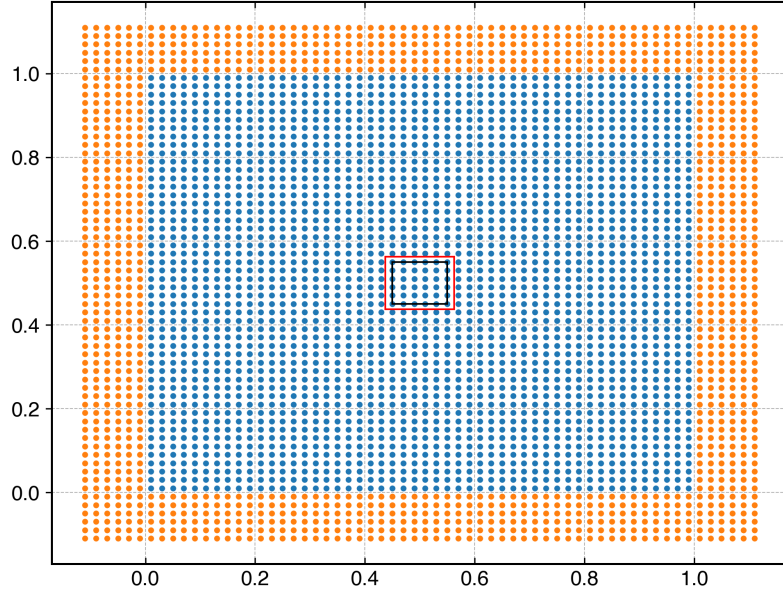


Figure 3.16 : The domain filled by blue fluid particles. The red box shows the smallest domain considered for the highest resolution of 8000×8000 and the black box shows the area which is considered to evaluate error for all the resolutions.

We first consider the MS given in eq. (3.9). We obtain the source term and solve eq. (3.4) using the L-IPST-C scheme for all the resolutions with $\nu = .01m^2/s$. We also consider the case where we do not correct the kernel gradient in the discretization of eq. (3.4) in the L-IPST-C scheme.

In fig. 3.17, we plot the L_∞ error in pressure and velocity solved using L-IPST-C scheme with kernel gradient corrected, after 100 timesteps as a function of resolution for $h_{\Delta x} = 1.2$ and $h_{\Delta x} = 1.4$. Clearly, we obtain second-order convergence. In fig. 3.18, we plot the error for the case where we do not employ kernel gradient correction. Clearly, the discretization error dominates.

We also consider the MS containing a range of frequencies given by

$$\begin{aligned}
 u(x, y, t) &= y^2 e^{-10t} \sum_{j=1}^{10} \sin(2j\pi x) \cos(2j\pi y), \\
 v(x, y, t) &= -e^{-10t} \sum_{j=1}^{10} \sin(2j\pi y) \cos(2j\pi x), \\
 p(x, y, t) &= e^{-10t} \sum_{j=1}^{10} \cos(4j\pi x) + \cos(4j\pi y).
 \end{aligned} \tag{3.18}$$

We obtain the sources terms and simulate eq. (2.14) using the L-IPST-C scheme for the above MS. As before, we also consider the case where we do not employ kernel gradient correction.

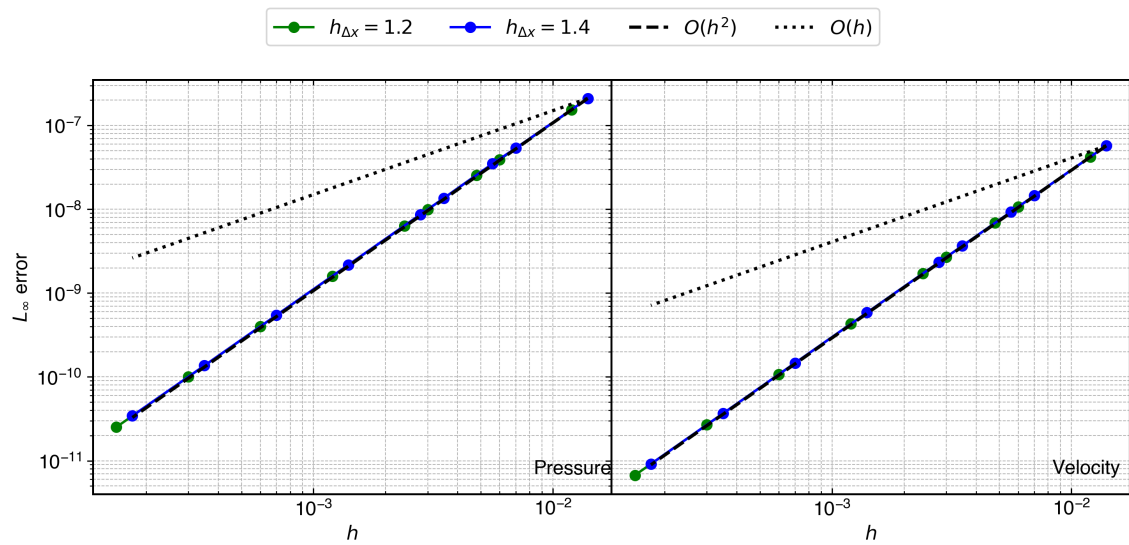


Figure 3.17 : The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.9). All cases are solved using the L-IPST-C scheme with kernel gradient correction.

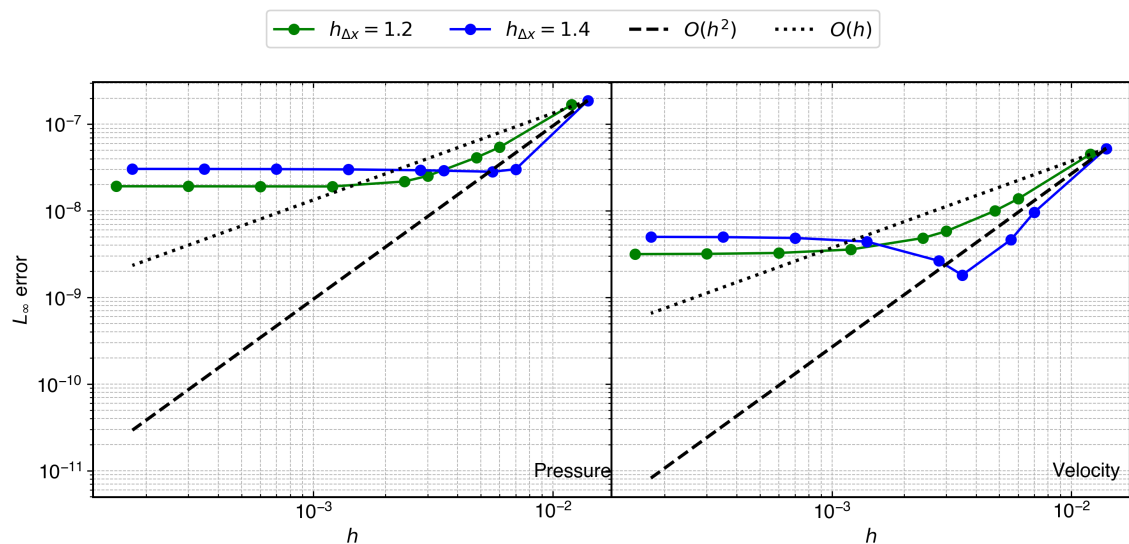


Figure 3.18 : The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.9). All cases are solved using the L-IPST-C scheme with no kernel gradient correction.

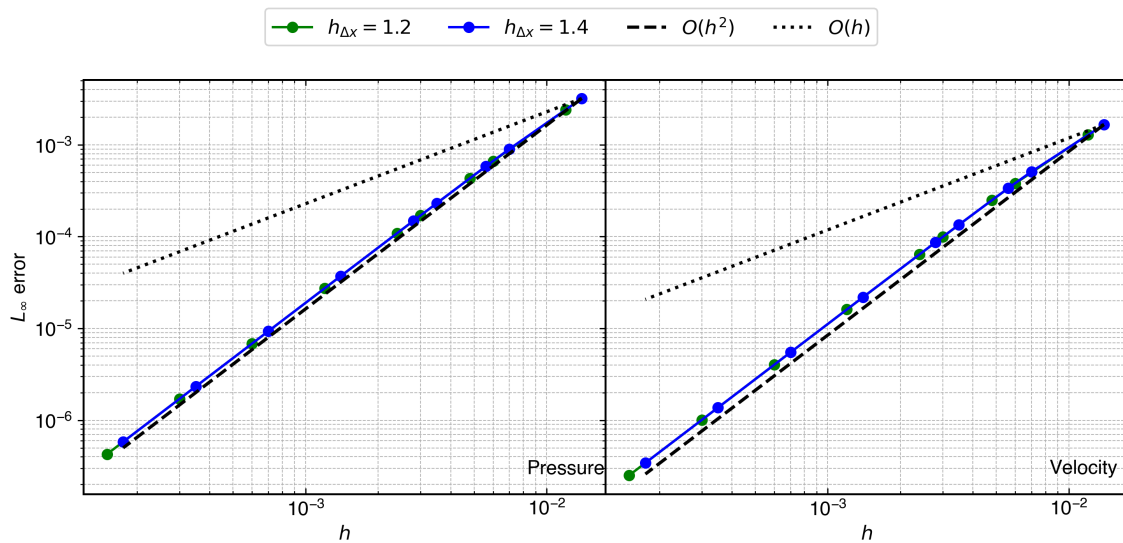


Figure 3.19 : The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.18). All cases are solved using the L-IPST-C scheme with kernel gradient correction.

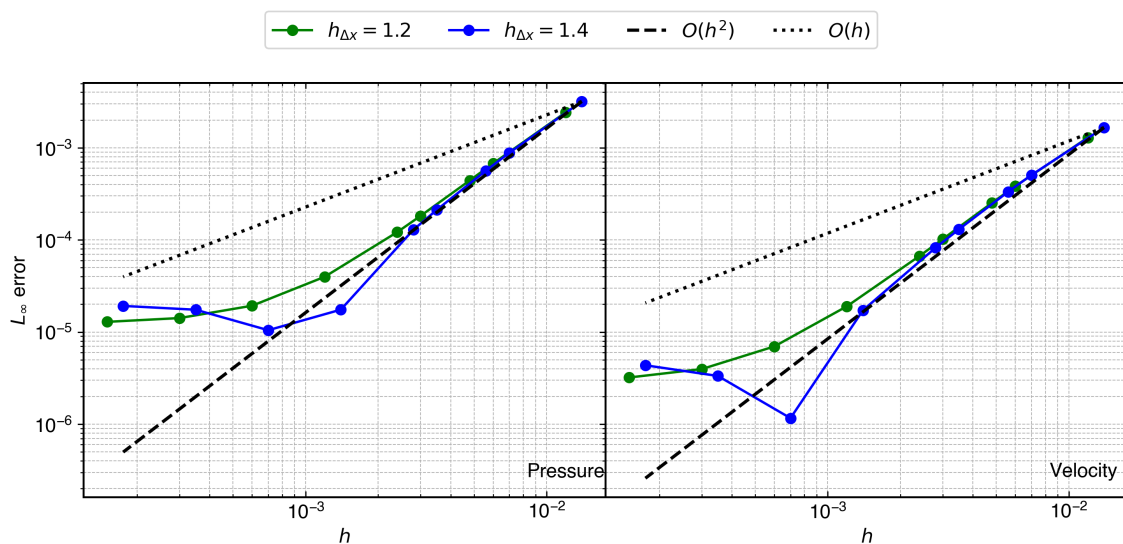


Figure 3.20 : The error in pressure (left) and velocity (right) as a function of resolution for two different $h_{\Delta x}$ values with the MS in eq. (3.18). All cases are solved using the L-IPST-C scheme with no kernel gradient correction.

In fig. 3.19, we plot the error in pressure and velocity solved using L-IPST-C scheme with kernel gradient correction for 100 timesteps as a function of resolution. Clearly, both cases show second-order convergence. In fig. 3.20, we plot the error in pressure and velocity for the solution obtained using the L-IPST-C scheme with no kernel gradient correction. As can be seen, kernel correction is essential in order to obtain second-order convergence at high resolutions.

We have therefore shown that we can consider very high resolutions using the MMS technique. This enables us to find flaws in the scheme, which may not converge at a very high resolution. These are hard to test using traditional methods where a fluid flow problem is solved.

3.3.7 Verification in 3D

We now use the MMS to verify the code with a three-dimensional domain. Since the number of particles in three dimensions increase much faster than in two dimensions with the increase in resolution, we can reduce the domain size with resolution as done while dealing with extreme resolutions. We consider a unit cube domain size with 1 million particles. As we increase the resolution, we decrease the size of the domain such that the number of particles in the domain remain at 1 million. We consider the MS given by

$$\begin{aligned}
 u(x, y, z, t) &= y^2 e^{-10t} \sin(\pi(2x + 2z)) \cos(\pi(2x + 2y)), \\
 v(x, y, z, t) &= -e^{-10t} \sin(\pi(2y + 2z)) \cos(\pi(2x + 2y)), \\
 w(x, y, z, t) &= -e^{-10t} \sin(\pi(2x + 2z)) \cos(\pi(2y + 2z)), \\
 p(x, y, z, t) &= (\cos(\pi(4x + 4y)) + \cos(\pi(4x + 4z))) e^{-10t}.
 \end{aligned} \tag{3.19}$$

We obtain the source term by subjecting the MS in eq. (3.19) to the governing equation in eq. (2.14) with $\nu = 0.01m^2/s$. We simulate the problem for 10 timesteps.

In fig. 3.21, we plot the L_∞ error in pressure and velocity as a function of resolution for the L-IPST-C scheme with and without kernel gradient correction. As expected, the case with no kernel gradient correction gradually flattened due dominance of discretization error. The case with kernel gradient correction shows second-order convergence in both pressure and velocity. Thus we see that we can easily test the SPH method in a three-dimensional domain using the MMS.

3.4 Summary

In this chapter, we have used the MMS to verify the convergence of different WSPH schemes. Thus far, most of the numerical studies of the accuracy and convergence of the

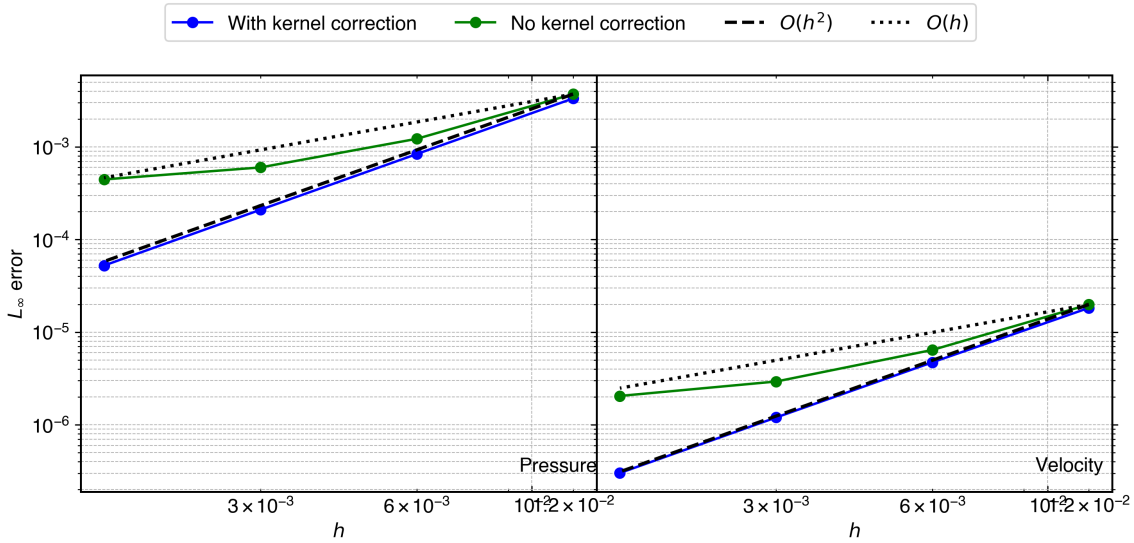


Figure 3.21 : The L_∞ error in pressure (left) and velocity (right) after 10 timesteps as a function of resolution solved using L-IPST-C scheme with and without kernel gradient correction. The source term is calculated using the MS in eq. (3.19).

WCSPH method have used either an exact solution like the Taylor-Green vortex problem or an established solver or experimental result. These methods are therefore limited in their ability to detect specific problems in an SPH implementation. This is true even in work discussed in the previous chapter, where a Taylor-Green vortex, Gresho-Chan vortex, and incompressible shear layer problems are used. These are complex problems, and obtaining a solution to these involves significant computation. Moreover, if the results do not produce the expected accuracy or convergence, the researcher does not obtain much insight into the origin of the problem. Furthermore, the conventional approaches do not offer any means to study the accuracy of boundary condition implementations.

In this context, the proposed approach offers a multitude of advantages listed and discussed below:

- The method is highly efficient in terms of execution time. We are able to detect problems in the implementations of specific discretization operators in less than 100 iterations. Even for our most challenging cases with a million particles, the typical run time for a single computation on a multi-core CPU does not exceed a few minutes. On the other hand, the comparison study for the lid-driven cavity case in section 3.1 took 150 minutes for just the 200×200 resolution.

- The method easily works in three dimensions, and we demonstrate its applicability for a simple three-dimensional case. This is significant because traditional SPH verification methods only use two-dimensional problems.
- The method allows us to identify particular problems with a code. Through a judicious choice of MS and time integrator, we can identify if the implementation of a specific term in the governing equation is the source of a problem. We have demonstrated this with several examples in the preceding sections.
- We are able to verify the order of convergence efficiently even for very high resolutions and thereby test if the scheme is truly second-order convergent as the resolution increases. In the present work, we have demonstrated this for extremely high resolutions (corresponding to 8000×8000 particles in a $1m \times 1m$ domain) without needing to simulate the problem for a long duration and also limiting the number of computational particles to a smaller number.
- The method will work on any manufactured solution, and this allows us to test the scheme with functions involving a large range of frequencies. In contrast, many exact solutions involve simple functional forms. Therefore by using the MMS, the code can be tested with a more challenging class of problems.

As a result of these significant advantages, the proposed method offers a robust, efficient, and powerful method to verify the accuracy and convergence of SPH schemes. We can also effectively test the boundary condition implementations using the MMS. However, in order to effectively test boundary condition implementations, it is important to capture features of the solid bodies accurately. In the next chapter, we discuss various methods to create initial particle distribution that accurately represents the object of interest.

Chapter 4

Construction of solid bodies in SPH

In the SPH method, solid, inlet, and outlet are usually represented by a few layers of dummy particles to implement boundary conditions. The fluid properties of these dummy particles, viz. velocity, pressure, and density, are extrapolated using different methods. However, to accurately implement the boundary conditions, one needs to capture the boundary surface adequately. The arrangement of particles in and around a solid body of interest is termed as *particle packing*. In this chapter, we discuss various existing particle packing techniques and their drawbacks, followed by a novel hybrid algorithm. Two widely used methods by Colagrossi et al. (2012) and Jiang et al. (2015) focus on the particles in either the fluid or solid domain only. Therefore, we propose a hybrid algorithm to pack the fluid and solid particles around the interface such that the geometric features are captured and the particles are uniformly distributed. We use the proposed algorithm to generate test cases for solid boundary condition implementation verification. In the next section, we discuss various packing algorithms in detail.

4.1 Particle packing algorithms

In a simulation with a solid body, for example, the flow past a circular cylinder in two dimensions, the solid particles are represented using dummy particles. The use of the particles arranged on a rectangular lattice would be an easy choice. However, the resulting boundary will be jagged. Furthermore, the accurate approximation of function and its derivatives requires uniform distribution of particles. Colagrossi et al. (2012) proposed a measure of uniformity of particle distribution. A distribution is said to be uniform such that the condition

$$\psi_i = \sum m_j W_{ij} = C_1, \quad (4.1)$$

and

$$\nabla\psi_i = \sum m_j \nabla W_{ij} = C_2, \quad (4.2)$$

are satisfied. For unit mass, $C_1 \approx 1$ and $C_2 \approx 0$. In order to produce body-conforming initial particle distribution, various particle packing algorithms are discussed in detail in the following sections.

4.1.1 Standard packing

In this method, we construct the interior of a 2D object using the method proposed by Marrone et al. (2011). We represent the object boundary by a piecewise linear curve (PLC) with normals to the boundary pointing out of the solid. We generate the first layer of the interior by moving the PLC points into the body along the normal by $\Delta x/2$ where Δx is the particle spacing. We discretize the new PLC into particles such that each particle is approximately Δx distance apart along the PLC. We move the newly added PLC further into the body along the normal by Δx and discretize again. We repeat this procedure until we generate the desired number of layers of solid particles. We note that this works only for 2D objects.

Once we create the dummy layers representing the solid body, we place fluid particles around the solid particles. We use the method proposed by Colagrossi et al. (2012) to pack particles around the fixed solid particles. In order to initialize the particle position, we consider a grid of evenly distributed particles and retain only the particles outside (defined by the direction of normal) the boundary surface represented by the PLC. A force due to number density gradient similar to eq. (4.2) packs the particles. We note that since we use only the number density gradient as a repulsion force amongst the particles, they are prone to clumping (Morris, 1995; Swegle et al., 1995). We subject the particles to a damping force to dissipate the energy of the system. Hence the force on any particle is governed by

$$\frac{d\mathbf{u}}{dt} = -\frac{\nabla p_b}{\psi} + \zeta \mathbf{u}. \quad (4.3)$$

The value of p_b and ζ is set directly (see section 4.1.3). We discretize the equation as

$$\frac{d\mathbf{u}_i}{dt} = p_b \sum_j \frac{\omega_i \omega_j}{m_j} \nabla W_{ij} - \zeta \mathbf{u}_i. \quad (4.4)$$

The convergence criteria will be discussed in section 4.1.3.

In algorithm 1, we describe the Standard packing in detail. The **ReadInput** function reads the points describing the geometry. We initialize all the particles, and the dummy particles are created using the method proposed by Marrone et al. (2011) in **CreateParticles**. The **SetConstantAndTimeStep** function sets the constants and time

Algorithm 1: Standard particle packing algorithm.

Result: Coordinates of solids and fluids

```

ReadInput();
CreateParticles();
SetConstantAndTimeStep();
converged = False;
iteration = 0;
while not converged do
    UpdateNeighbors();
    ComputeAccelerations();
    IntegrateParticles();
    converged = CheckConvergence();
    iteration++;
end

```

step (see section 4.1.3). The iteration starts with the creation of neighbor lists for every particle in **UpdateNeighbors**. Then, we compute accelerations in **ComputeAccelerations** using eq. (4.3) and integrate in **IntegrateParticles** using eq. (4.17). The iteration continues until we satisfy the criteria (see section 4.1.3) in **CheckConvergence**.

4.1.2 Coupled packing

Jiang et al. (2015) proposed a packing algorithm for solid objects both in 2D and 3D in order to sample blue noise. We describe the steps involved in algorithm 2. We use a repulsion force which is similar to the one used in Colagrossi et al. (2012) along with damping. However, in this case, we use a symmetric form to discretize the background pressure force, given by

$$\mathbf{a}_{b,i} = -m_i p_b \sum_j m_j \left(\frac{1}{\psi_i^2} + \frac{1}{\psi_j^2} \right) \nabla_i W_{ij}, \quad (4.5)$$

where W_{ij} is the cubic spline kernel function. We compute this in **ComputeAccelerations** along with an additional force discussed later. In the present implementation, a constant background pressure p_b is used. In the original method, the background pressure is a function of particle density. On computing the acceleration, we integrate all the particles in **IntegrateParticles**.

Since the particles near the surface lack supporting particles, a large force acts upon them. In order to keep the particles inside a confined region, we convert the particles

nearer than $0.05\Delta x$ to boundary particles. The motion of these particles on the boundary is discussed in the next section. As done in the original method, we project the boundary particles back to the surface in every iteration in **ProjectParticles**.

Algorithm 2: Packing algorithm by Jiang et al. (2015).

```

Input: particles, max_iter
Result: Packed particles
iteration = 0;
while not converged and iteration < max_iter do
    UpdateNeighbors(particles);
    ComputeAccelerations();
    IntegrateParticles();
    ProjectParticles();
    iteration++;
end

```

If one were to only use eq. (4.5), it would result in more particles pushed towards the boundary. In order to counteract the force on the particles near the boundary, Jiang et al. (2015) used a cohesion force proposed by Akinici et al. (2013). The acceleration due to this force in SPH form is given by

$$\mathbf{a}_{c,i} = -m_i\gamma \sum_j m_j\psi_{ij}\mathcal{C}_{ij}\hat{\mathbf{n}}_{ij}, \quad (4.6)$$

where $\psi_{ij} = 2\psi_o/(\psi_i + \psi_j)$, $\hat{\mathbf{n}}_{ij} = \mathbf{x}_{ij}/r_{ij}$ and \mathcal{C}_{ij} is the spline kernel in Akinici et al. (2013), given by

$$\mathcal{C}(q) = \frac{32}{\pi h^d} \begin{cases} (1-q)^3 q^3 & 0.5 < q < 1, \\ 2(1-q)^3 q^3 - \frac{1}{64} & 0 < q < 0.5, \\ 0 & \text{otherwise.} \end{cases} \quad (4.7)$$

Jiang et al. (2015) have not specified a way to choose the value of the constants in eq. (4.5) and eq. (4.6). We heuristically choose the values of $\gamma = 20$, $p_b = 10$ and $h = \Delta x$ for all the simulations.

We note that there is no exterior defined in the original method of Jiang et al. (2015). In this work, we pack the exterior in the same manner as the interior by moving the boundary surface along the surface normal by $\Delta x/2$ and $-\Delta x/2$ for the exterior and interior, respectively. In fig. 4.1, we enclose the interior domain (domain 1) within a thick dashed line, and the exterior is within a thin dashed line (domain 2). We represent each domain with a different pattern. Domain 2, having an external boundary, has frozen particles outside it. The solid line represents the boundary surface.

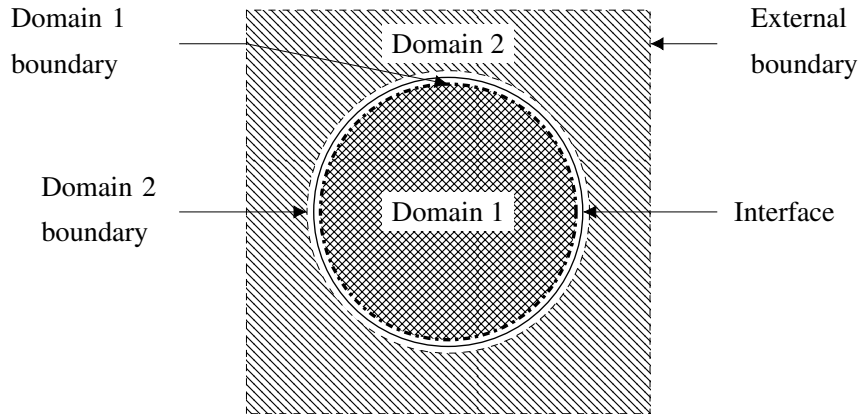


Figure 4.1 : Schematic for the Coupled packing algorithm. The external region is marked as domain 2, and the internal region is marked as domain 1.

Algorithm 3: Coupled particle packing algorithm.

Result: Coordinates of solids and fluids

```

ReadInput();
DivideDomain();
CreateParticles();
SetConstantAndTimeStep();
Algorithm2(domain1, 5000);
Algorithm2(domain2, 5000);
iteration = 0;
converged = False;
while not converged do
    UpdateNeighbors(domain1 + domain2);
    ComputeAccelerations();
    IntegrateParticles();
    converged = CheckConvergence();
    iteration++;
end
SeparateParticles();

```

We pack the particles in two different passes as described in algorithm 3. First, we read the geometry data in **ReadInput** followed by particle initialization in **CreateParticles**. We divide the free particles into domain 1 and domain 2 depending upon their location in **DivideDomain**. We do the following for particles that are lying in between the two dashed boundaries of domains 1 and 2. If a particle is in the exterior region (outside the boundary surface), we move it along the normal by a distance Δx into

domain 2. Similarly, we move particles between the boundary surface and the domain 1 boundary into domain 1.

In the first pass, we solve domains 1 and 2 separately using the algorithm 2. In this case, each domain is unaware of the other. We perform the boundary particle projection onto the open surface of the respective boundaries. We move the particles in both domains for a predetermined number of iterations, at which point the particles reach equilibrium¹. This step ensures that we sort all particles as either interior or exterior and have an interface that they cannot cross, i.e. the dashed lines.

In the second pass, when the projection is complete, we constrain the particles on the interior surface, i.e. the thick dashed line, to move along it. We allow all other particles to freely move using eq. (4.5) in **ComputeAccelerations** and **IntegrateParticles**. The iteration continues until the convergence criteria in **CheckConvergence** as discussed in section 4.1.3 is satisfied. The presence of exterior particles eliminates the need for the cohesion force; thus, we do not evaluate it once domains 1 and 2 start interacting. Using this approach, we obtain a uniform distribution both inside and outside the surface. We note that the original implementation was used to sample blue noise and does not require external particles.

In both algorithms discussed above, only one of the fluid or solid particles were packed at a time. In the next section, we propose a new hybrid algorithm, where we simultaneously pack solid and fluid particles around an arbitrarily-shaped body.

4.1.3 Hybrid packing

The schematic shown in fig. 4.2 depicts the different kinds of particles used in the proposed algorithm. In the figure, we consider the two-dimensional case of a circular cylinder surrounded by fluid. The dashed black line represents the surface of the cylinder (boundary surface), which we wish to capture accurately. We assume that this surface is discretized into a set of points called “boundary nodes”. The different kinds of entities shown in the figure are,

- *Free particles*: These are particles arranged initially in a rectangular or hexagonal-packed pattern. Their motion is not constrained. These are depicted as blue circles.
- *Frozen particles*: These are a set of fixed dummy particles that surround the free particles in order to provide support to the kernel. These are depicted as green circles.

¹We choose the predetermined iterations to be 5000 based on the test cases considered.

- *Boundary particles*: These particles are constrained to move along the “boundary surface” and are depicted as red circles.
- *Boundary surface*: The surface of the geometry that is discretized. It is represented by a set of fixed points called “boundary nodes” which do not influence any other particles.
- *Boundary node*: These are points that discretize the boundary surface; they also store the local surface normals of the boundary surface. These are depicted as black dashes. We note that these are called nodes because they do not exert any forces on particles and only serve to provide information on the position and orientation of the boundary surface.

During the execution of the algorithm, free particles (blue) may be converted to boundary particles (red) if they are close to the boundary surface. Once the proposed algorithm completes, the red boundary particles must conform to the boundary represented by the dashed black line. The blue particles inside the dashed boundary will be considered solid (dummy) particles, and those outside as fluid particles.

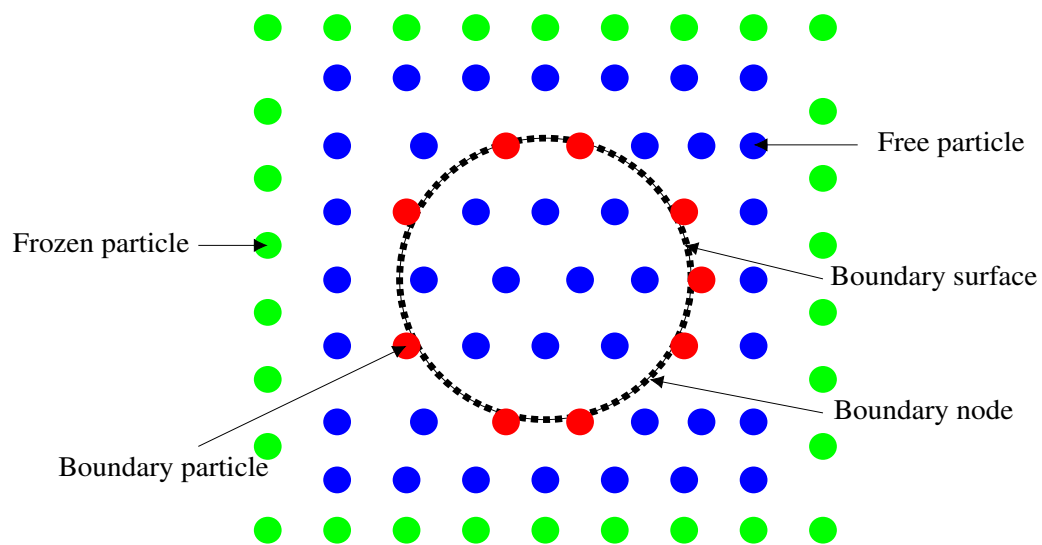


Figure 4.2 : Schematic of the initial distribution of particles and the different kinds of particles.

In fig. 4.3, we show the overall flow of the algorithm. The algorithm requires two inputs viz. the geometry information and the desired particle spacing. Given the geometry surface, we first estimate the number of particles N_s that should lie on this surface using the desired spacing of particles and the length of the surface in 2D (the surface area in 3D).

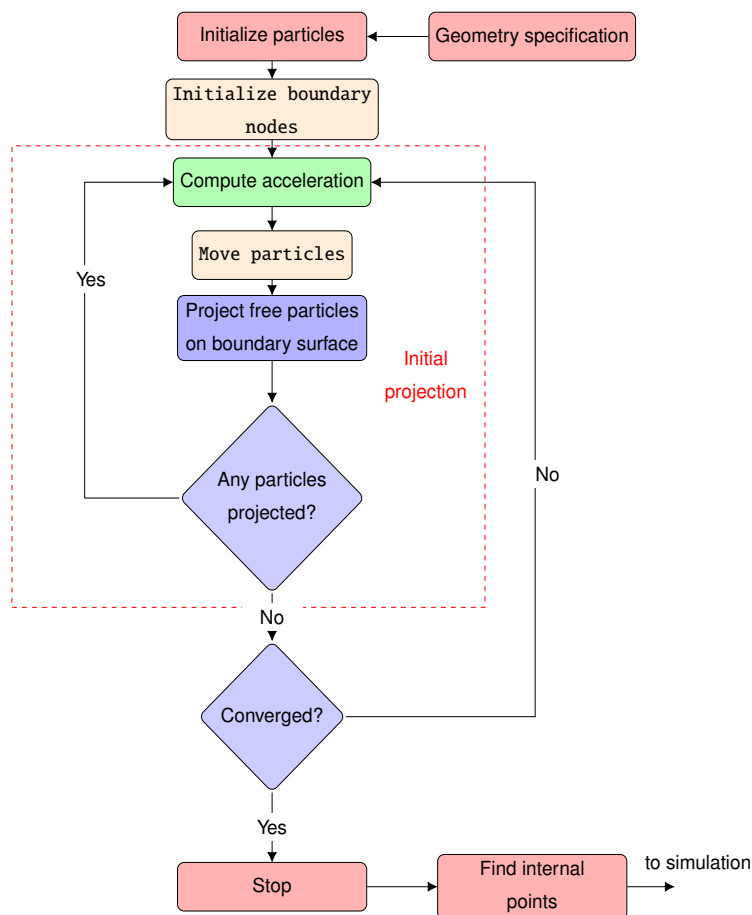


Figure 4.3 : Flowchart of the particle packing algorithm. The box outlined in dashed red lines is the initial projection phase.

Initially, we create the frozen particles on the periphery of the domain placed on a rectangular lattice as shown in fig. 4.2. Then, we place the free particles inside the domain on a regular lattice. At this stage, we do not identify the boundary particles. We initialize the boundary nodes using the information provided by the user.

We compute the acceleration on the free and boundary particles (we identify these later) using a local density gradient and a repulsive force. The last step corresponds to the green block in fig. 4.3. We move the particles using the computed accelerations. As the free particles move, we convert them to boundary particles if they are close enough to the boundary surface. We project them to the nearest point on the boundary surface. The boundary particles only move along the boundary surface. We convert the free particles iteratively to boundary particles until no free particle is sufficiently close to the boundary.

During the initial projection phase (denoted by the red dashed line in the fig. 4.3), we project the particles regularly onto the boundary surface. We repeat the last step until the number of boundary particles has reached N_s and remains there for a few consecutive iterations. The algorithm then proceeds to settle the particles into a uniform distribution

until the displacement of the particles is less than a user-defined tolerance. We denote this step as the “Converged” block in fig. 4.3. Once we attain convergence, we obtain the boundary and free particles packed inside and outside the surface as desired. Since the boundary surface is known, we can quickly identify the free particles as solid and fluid particles.

We can place this packed collection of free and boundary particles into a larger regular mesh of particles for a simulation. For example, In fig. 4.4 the dashed region shows where we can place the packed particles. We use a regular mesh of the same spacing to represent the exterior of this region (shown in green). This approach is convenient to use in the context of fluid flow past solid bodies as done for internal flows (Negi et al., 2021a), and free surface flows (Tafuni et al., 2018). We note that the particle packing makes the particle unstructured, whereas, in the case of a mesh-based solver, a structured arrangement of cells is preferred near a solid wall. However, in the SPH-based solver, the disorder in the particle distribution over time is inevitable (unless a Eulerian method is employed). Therefore, in the proposed method, we lose the structured nature of the initial particle distribution and choose unstructured initial particle distribution that accurately captures the solid boundary features. In the subsequent sections, we explain the algorithm described above in detail.

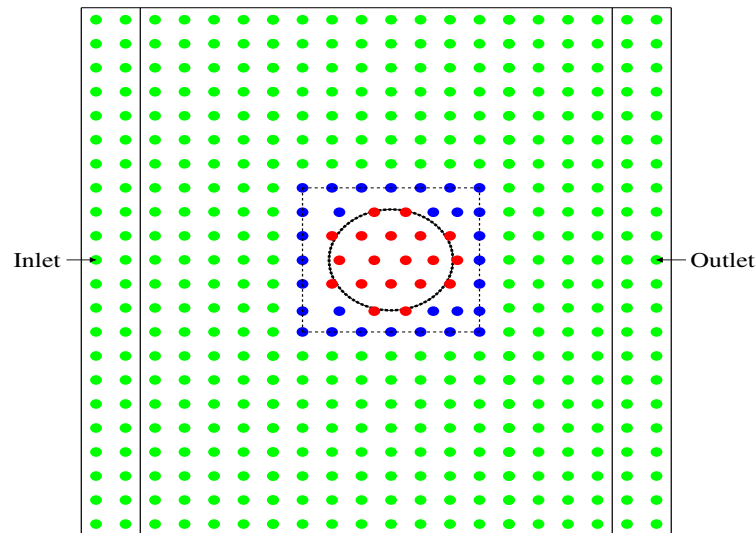


Figure 4.4 : The pre-processed patch (within dashed lines) of particles placed in the appropriate location of a typical simulation. The blue particles denote the free particles identified as fluid particles, and the red represents the solid particles identified by the packing process. The green particles are generated from a fixed mesh of points.

Initialization of boundary nodes

We first initialize the boundary nodes, which represent the boundary surface. In a two-dimensional domain, we require a set of points that discretize the boundary curve. We parametrize the boundary curve by λ , and specify the points on the curve as $\mathbf{x} = C(\lambda)$ and $\lambda \in [0, 1]$. We discretize this curve such that $\mathbf{x}_i = C(\lambda_i)$. We keep the spacing between points such that $|\mathbf{x}_{i+1} - \mathbf{x}_i| < \Delta x$. We initialize the boundary node coordinates using these points. We calculate the outward normals $\hat{n}_{x,i}, \hat{n}_{y,i}$ for any node i using

$$\begin{aligned}\hat{n}_{x,i} &= 0.5 \left(\frac{y_{i+1} - y_i}{r_{i+1\ i}} + \frac{y_i - y_{i-1}}{r_{i\ i-1}} \right), \\ \hat{n}_{y,i} &= -0.5 \left(\frac{x_{i+1} - x_i}{r_{i+1\ i}} + \frac{x_i - x_{i-1}}{r_{i\ i-1}} \right),\end{aligned}\tag{4.8}$$

where r_{ij} is the length of the segment joining the node at (x_i, y_i) with the node at (x_j, y_j) . We normalize the resulting normal. Equation (4.8) ensures that sharp corners of the curve have smooth normals. For a three-dimensional case, a triangulation of the surface with outward normals are necessary. One can generate the triangulation using any mesh generation tool (Bern et al., 1992). We use the centroid of each triangle and its normal to initialize the boundary nodes.

In SPH, the actual boundary surface is exactly in between the solid and fluid particles. Thus, both in two and three dimensions, given a particle spacing of Δx , we shift the boundary nodes by $\Delta x/2$ inside the actual boundary to correctly implement the solid boundary conditions as discussed in Marrone et al. (2011). In order to move the nodes inwards, we perform the following translation on each boundary node given by

$$\mathbf{x} = \mathbf{x} - \frac{\Delta x}{2} \hat{\mathbf{n}},\tag{4.9}$$

where, \mathbf{x} is the position of the node and $\hat{\mathbf{n}}$ is its unit normal pointing outwards. We must note that this is optional, and one can provide a pre-shifted surface and avoid the shifting with eq. (4.9).

One must take care when there are sharp changes in the features of the geometry. Consider an airfoil trailing edge in fig. 4.5 shown as a black line. We mark the sharp corners as ‘‘corner nodes’’, and in this case, it is the i^{th} node. When we shift the nodes on this surface, the boundary surface tends to self-intersect itself. This is shown by the red nodes connected using a black dashed line. In order to remove the intersection of the boundary surface near the corner node, we replace the points $i - 3$ to $i - 1$ with the points on the line joining $i - 4$ and i with equal spacing shown by black points. Similarly, we also replace the points $i + 1$ to $i + 3$ with points lying along the line joining points i and $i + 4$. This results in a non-intersecting surface, as shown by the red dashed line on the right

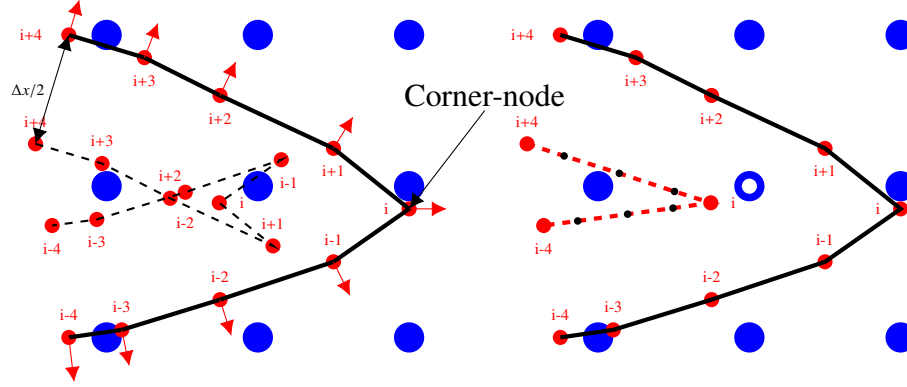


Figure 4.5 : Shifting of the boundary near a sharp-edged boundary. The boundary nodes are depicted in red with normals. On the left is the boundary surface after the initial shifting shown as black dashed lines. On the right is the final geometry after the removal of the intersecting edges, which are shown as dashed red lines. The annular blue free particle is the candidate to be placed on the corner and held fixed.

side of fig. 4.5. Once we resolve the intersection, we place the nearest free particle near the corner node (annular blue particle) on it and convert it to a *fixed boundary particle*. The position of these fixed boundary particles does not change in the entire simulation.

In the case of a three-dimensional object, one has to make sure that the surface does not intersect after applying eq. (4.9) or use a pre-shifted surface as an input.

Dynamics of the particles

In this section, we discuss the dynamics of particle regularization. We apply two forces to the particles, and together these regularize the particle distribution. The two forces are a gradient due to particle disorder and a pure inter-particle repulsive force.

In the presence of a constant pressure field p_b and no viscous effect, the momentum equation is given by

$$\frac{d\mathbf{u}}{dt} = -\frac{\nabla(1 \cdot p_b)}{\psi} = -\frac{p_b \nabla(1) + \nabla(p_b)}{\psi}. \quad (4.10)$$

When the term, $p_b \nabla(1)$, on the right hand side is discretized using the SPH method, we obtain $p_b \nabla \psi$ (see eq. (4.2)) for unit mass. This is non-zero when the particles are not uniform and hence particles exert a force on each other in order to reach an equilibrium position. Using the SPH approximation, we discretize the above equation as

$$\mathbf{a}_{b,i} = \frac{d\mathbf{u}_i}{dt} = -p_b \sum_j \frac{\omega_i \omega_j}{m_i} \nabla W_{ij}. \quad (4.11)$$

Since all SPH kernels satisfy eq. (4.1), any suitable SPH kernel discussed in section 2.1 can be employed. In this work, we use the quintic spline kernel.

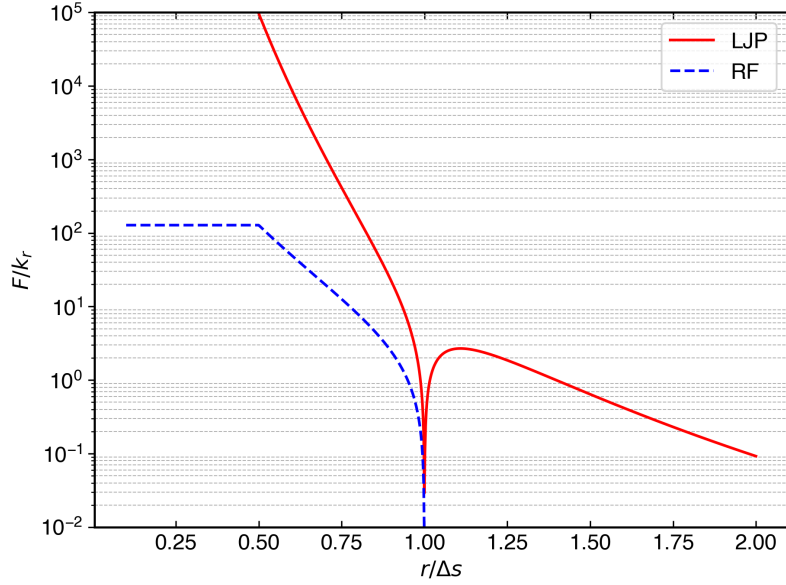


Figure 4.6 : Force due to LJ potential and the gradient of eq. (4.12) as a function of distance, r and $\alpha = 1$.

In this algorithm, the value of p_b is set to 1 independent of the resolution. In addition to this force, we use a repulsive force (RF) similar to the gradient of the Lennard Jones potential (LJP). The new repulsion force potential (ϕ_{RF}) is given by

$$\phi_{RF} = 12 k_r \left(\frac{\tau^2}{r^3} - \frac{\tau}{r^2} \right), \quad (4.12)$$

where k_r is a constant. We set $\tau = 2\kappa\Delta x/3$, where κ is a scaling factor. The gradient of eq. (4.12) gives us the force due to ϕ_{RF} . We keep the force constant for $r < \Delta x/2$ in order to avoid very large repulsion forces. We write the SPH approximation of the acceleration due to eq. (4.12) as

$$\mathbf{a}_{RF,i} = -\nabla\phi_{RF,i} = \begin{cases} \sum_j 192 k_r \mathbf{e}_{ij} \left(\frac{3\tau^2}{\Delta x^4} - \frac{\tau}{\Delta x^3} \right) & r_{ij} \leq \Delta x/2, \\ \sum_j 12 k_r \mathbf{e}_{ij} \left(\frac{3\tau^2}{r_{ij}^4} - \frac{2\tau}{r_{ij}^3} \right) & \Delta x/2 < r_{ij} \leq \alpha\Delta x, \\ 0 & r_{ij} > \alpha\Delta x, \end{cases} \quad (4.13)$$

where $\mathbf{e}_{ij} = \mathbf{x}_{ij}/r_{ij}$. Note that the acceleration is continuous at $r_{ij} = \Delta x/2$. We find that using a value of $\kappa = 0.95$ works well for the algorithm.

It is clear from eq. (4.13) that this force is active only when particles come closer than the desired particle spacing. This force prevents particle pairing, which may happen due to the use of some kernels like cubic spline for large time steps (Dehnen et al., 2012). In fig. 4.6, we show the comparison between the force due to LJP and RF. The LJP repulsion force increases rapidly compared to our suggested repulsion force. Thus, This force allows us to use a larger time step during integration. Moreover, unlike the force due to LJP, the new force does not introduce inter-particle attraction.

As discussed in Colagrossi et al. (2012), for stability, we use a damping force to reduce the energy of the system. We compute the acceleration due to damping for the i^{th} particle by

$$\mathbf{a}_{d,i} = -\zeta \mathbf{u}_i, \quad (4.14)$$

where ζ is the damping constant and \mathbf{u}_i is the velocity of the i^{th} particle. We discuss the value of the damping constant ζ in section 4.1.3. Thus, the equation governing the dynamics of the system is

$$\frac{d\mathbf{u}}{dt} = -\frac{\nabla p_b}{\psi} - \nabla \phi_{RF} - \zeta \mathbf{u}. \quad (4.15)$$

We convert the above equation into SPH form using eqs. (4.11), (4.13) and (4.14); thus acceleration of the i^{th} particle

$$\mathbf{a}_i = \mathbf{a}_{b,i} + \mathbf{a}_{RF,i} + \mathbf{a}_{d,i}. \quad (4.16)$$

We must note that the combination of background pressure and repulsion forces produces repulsion only when particles are disordered. One can also accomplish this by various other particle shifting techniques (PST) first proposed by Xu et al. (2009).

On using eq. (4.15), we calculate the velocities and new positions using a semi-implicit Euler integration given by

$$\begin{aligned} \mathbf{u}_i(t + \Delta t) &= \mathbf{u}_i(t) + \Delta t \mathbf{a}_i(t), \\ \mathbf{x}_i(t + \Delta t) &= \mathbf{x}_i(t) + \Delta t \mathbf{u}_i(t + \Delta t). \end{aligned} \quad (4.17)$$

In the case of boundary particles, we correct the velocities to constrain them to move along the surface (discussed later). We should note that the packing algorithm only ensures that particle distributions are regular, and therefore using a higher-order integrator would not promise better results.

We study the stability of the method in a two and three-dimensional domain for finite perturbation under the action of forces described above. We consider a domain of size $1m \times 1m$ unit along each coordinate direction. We place particles with a spacing of $\Delta x = 0.05$. We also place fixed particles outside this unit box suitably. The criteria to set the value of p_b , ζ , k_r and Δt are discussed later. We perturb a single particle close to the center by $\Delta x/2$ in each direction. We move the particles using eq. (4.15) and eq. (4.17) for 15000 iterations. We consider the stability of the following initial distribution of particles:

- rectangular packing shown in fig. 4.7a with ND gradient.
- rectangular packing shown in fig. 4.7a with $ND + RF$.

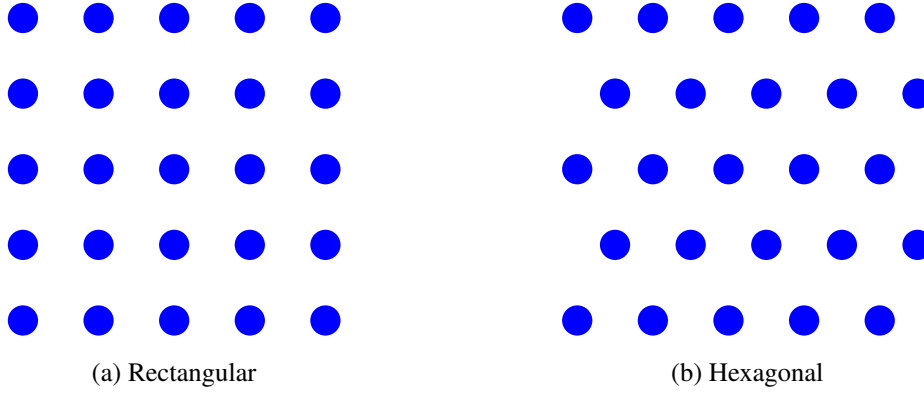


Figure 4.7 : Different packing structures in 2D.

- hexagonal packing shown in fig. 4.7b with number density (ND) gradient.
- hexagonal packing shown in fig. 4.7b with ND + repulsion force (RF).

We evaluate the error in every iteration using

$$L_\infty(\psi - \psi_o) = \max(|\psi_i - \psi_o|, \forall i \in [1, N]), \quad (4.18)$$

where N is the number of particles in the domain. In fig. 4.8a and fig. 4.8b, we plot the $L_\infty(\psi - \psi_o)$ with number of iterations for 2D and 3D domain respectively with $\psi_o = 1.0$. In 2D, all the combinations perform well. In the case of the 3D domain, rectangular lattices settle into an equilibrium configuration with a much larger density difference. Therefore the rectangular lattice in 3D is an unstable equilibrium. In contrast, the hexagonal packing in 3D is in a stable equilibrium.

We can understand the behavior in 3D from the total potential energy of the particles. We can see that eq. (4.1) for ψ_i is the potential energy of the i^{th} particle and hence $\sum_i \psi_i$ is the total potential energy of the system. The acceleration of a particle given in eq. (4.10) is the negative of the gradient of its potential energy. We have numerically found that a small perturbation of a particle at a given site reduces the total potential energy in the rectangular lattice case, whereas the same perturbation results in higher potential energy in the case of the hexagonal packing. It suggests that the hexagonal packing is stable in 3D, unlike the rectangular lattice. A careful analysis of the stability is outside the scope of the present work. Due to the stability of hexagonal packing shown in fig. 4.7b, we use it in all our test cases.

Projecting free particles to the surface

Initially, the boundary particles are unlikely to conform to the boundary surface. At $t = 0$, we assign no particles as boundary particles. We convert free particles to boundary

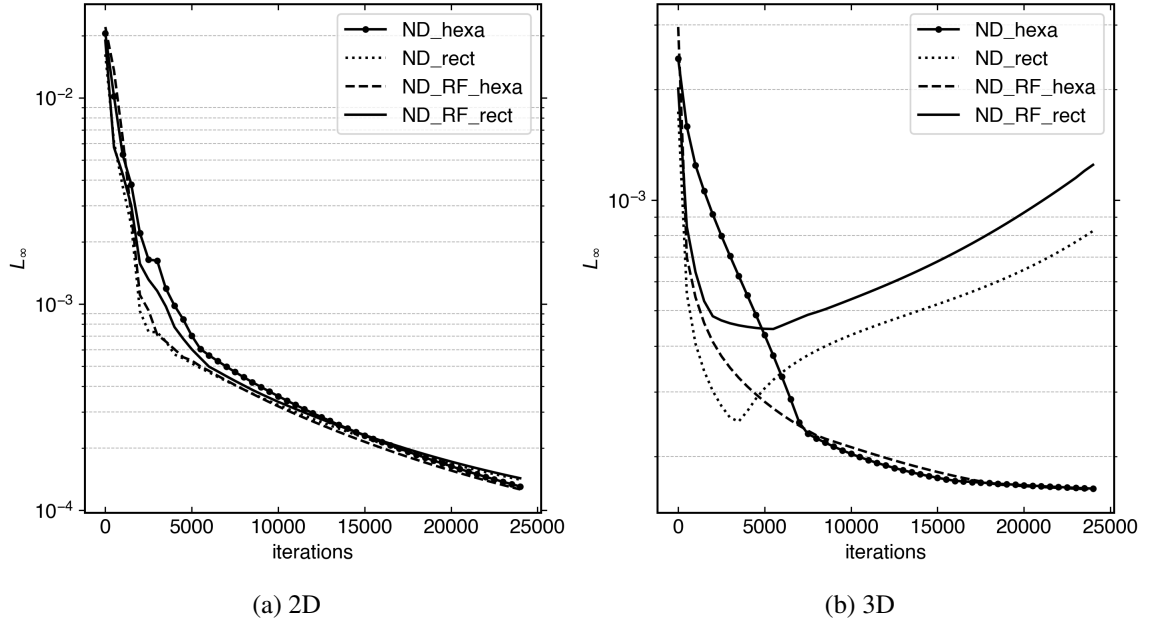


Figure 4.8 : Particle density convergence with a particle at center perturbed by $\Delta x/4$.

particles after every 50 iteration. We refer to this as the “projection frequency”. The projection frequency is a user-defined parameter. Increasing this does not guarantee better results; however, we do not recommend a very small value. On running the proposed algorithm with spacing $\Delta x = 0.1$ around a unit radius 2D cylinder with varying projection frequency as shown in fig. 4.9, we found that after projection frequency 20, the number density gradient does not change by a large value. Thus, we choose the value of 50 heuristically.

If we consider a flat surface, an estimate for the number of particles that can fill the surface is, $N_s = A_s/\Delta x^{(n-1)}$ where A_s is the area of the surface and n is the dimension of the space in which the surface is embedded. In order to perform projection, we employ two different criteria depending on whether the initial projection in fig. 4.3 is complete or not, as shown in algorithm 4.

In the algorithm, the function **FindParticlesNearBoundary** finds all the free particles that are less than a prescribed distance, **maxdist**, to a boundary node. We compute the distance as follows, given a free particle p , we find the boundary node b that is closest and compute the distance $\mathbf{r}_{pb} \cdot \hat{\mathbf{n}}_b$, where $\mathbf{r}_{pb} = \mathbf{r}_p - \mathbf{r}_b$ and $\hat{\mathbf{n}}_b$ is the normal at b . We note that the algorithm to find the nearest boundary node uses the existing neighbors generated when computing the accelerations discussed in section 4.1.3.

The **empty_count** variable stores the number of consecutive passes for which the **p_list** was empty. We convert the free particles having a distance to the boundary node less than $0.5\Delta x$ to boundary particles in **ConvertParticles** and project them to the bound-

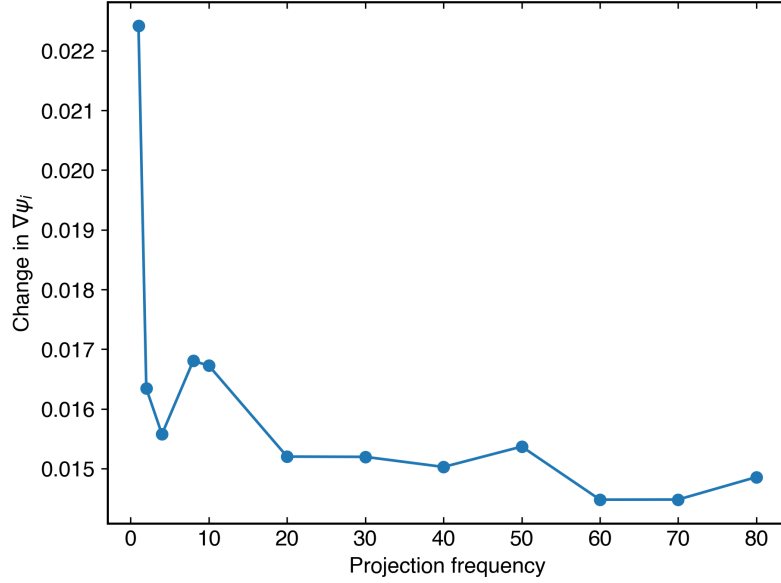


Figure 4.9 : Error in $\nabla\psi_i$ for a domain with unit radius cylinder with different projection frequencies.

ary surface in **ProjectToBoundary**. However, if the **empty_count** exceeds 3, we increase the distance threshold to $0.65\Delta x$. We increase the threshold to handle the corner cases where free particles are just outside $0.5\Delta x$ distance². This iterative conversion of free particles to boundary particles is necessary in order to capture the surface accurately.

Kinematics of boundary particles

As discussed earlier, we constrain the movement of the boundary particles along the boundary surface. Figure 4.10 illustrates the motion of the boundary particles (in red) along nodes 1, 2, 3, 4, and 5, representing the geometry. We note that even though the particle is a boundary particle, it is only projected onto the surface every **proj_freq** timesteps as discussed in section 4.1.3; hence the boundary particle may not exactly lie on the boundary as it moves.

We perform the motion of the boundary particle as follows. We first identify the boundary node nearest to the boundary particles in the direction of the particle's velocity. Consider a boundary particle p , and a boundary node j , near p having position \mathbf{x}_p and \mathbf{x}_j respectively. We determine the index of the nearest node J by

$$J = \arg \min_j \{r_{pj} \mid j \in \mathcal{N} \text{ and } \mathbf{x}_{pj} \cdot \mathbf{u}_p < 0\}, \quad (4.19)$$

²We find this value based on numerical experiments. Too small a value like $0.55\Delta x$ produces poor results in coarse resolutions, and too large a value ($0.75\Delta x$) produces poor distributions with finer resolutions.

Algorithm 4: Pseudo-code for free particle projection.

```

Input: proj_freq, ds= $\Delta x$ 
Result: List of free particles to be projected
iteration = 0;
empty_count = 0;
while not converged do
    ...;
    if iteration % proj_freq == 0 then
        if empty_count < 4 then
            p_list = FindParticlesNearBoundary(maxdist=0.5*ds);
            if len(p_list) > 0 then
                ConvertParticles();
                empty_count = 0;
            else
                empty_count++;
            end
        else
            p_list = FindParticlesNearBoundary(maxdist=0.65*ds);
            ConvertParticles();
        end
        ProjectToBoundary();
        ...;
    end
    iteration++;
end

```

where $\mathbf{x}_{pj} = \mathbf{x}_p - \mathbf{x}_j$, $r_{pj} = |\mathbf{x}_{pj}|$, \mathbf{u}_p is the velocity of the boundary particle and \mathcal{N} is the set of neighboring nodes of p . In fig. 4.10, the node 2 satisfies the conditions in eq. (4.19), and so $J = 2$. Using the nearest node index J , and the boundary particle p , the direction of motion $\mathbf{e}_r = -\mathbf{x}_{pJ}/|\mathbf{x}_{pJ}|$. Thus, we update the boundary particle position using the following equation

$$\mathbf{x}_p^{m+1} = \mathbf{x}_p^m + (\mathbf{u}_p^{m+1} \cdot \mathbf{e}_r^m) \mathbf{e}_r^m \Delta t, \quad (4.20)$$

for the m^{th} timestep.

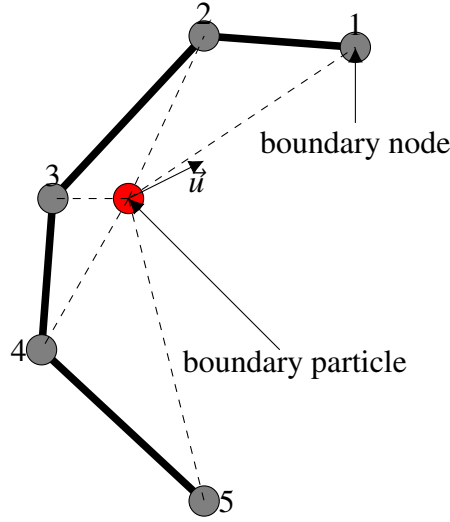


Figure 4.10 : Motion of boundary particle along the geometry.

Convergence criteria

For a perfectly packed distribution of particles, each particle should satisfy the condition in eq. (4.2) i.e. $\nabla\psi_i = 0$. However, this would take a lot of computational time. In the case of geometries having irrational volumes like the unit circle (having the volume equal to π), one could never achieve a perfect convergence, given a fixed resolution. Thus, similar to Jiang et al. (2015), we use the following criteria for convergence

$$\frac{\max\{|\mathbf{u}_i| \forall i \in N\} \Delta t}{h} < \epsilon, \quad (4.21)$$

where $\epsilon = 10^{-4}$ is the tolerance for all our test cases, $|\mathbf{u}_i|$ is the velocity magnitude of i^{th} particle, and we take the maximum over all the particles.

Determining the constants and timestep

It is important to choose the parameters ζ and k_r appropriately. We observe that eq. (4.11) scales as $O(p_b \Delta x^{-1})$ and eq. (4.13) scales as $O(k_r \Delta x^{-2})$. By requiring that these forces be of the same order, we can derive the relation

$$\frac{k_r}{p_b} = C_k \Delta x, \quad (4.22)$$

where C_k is an arbitrary constant. To find a suitable value, we consider a 2D lattice of points. We then perturb a single particle by $\Delta x/4$ in each direction. We apply a similar procedure to a 3D lattice. We find that a value of C_k in the range 0.004 – 0.006 ensures that the two forces are of the same order.

In order to ensure stability, the damping constant has a form similar to the one suggested by Colagrossi et al. (2012) given by $\zeta = C_\zeta/\Delta x$, where C_ζ is set in the range 0.2 – 0.5 resulting in an underdamped system.

The system evolves in pseudo-time. The time step Δt is set as in Adami et al. (2013) given by,

$$\begin{aligned}\Delta t_{p_b} &= 0.1 \frac{h}{p_b}, \\ \Delta t_\zeta &= \sqrt{0.1 \frac{h}{\zeta U}}, \\ \Delta t &= \min(\Delta t_{p_b}, \Delta t_\zeta),\end{aligned}\tag{4.23}$$

where U is the velocity magnitude. We note that we have used a $p_b = 1$ for all our simulations.

Separating interior and exterior particles

At the end of the simulation, we obtain interior (particles inside the boundary surface) and exterior particles (particles outside the boundary surface) uniformly distributed. We extract and use the interior particles and the boundary particles as solid particles, while the rest of the particles as fluid particles. In order to detect interior and exterior particles, we adopt this simple SPH-based procedure:

1. Find the nearest boundary node j to free particle i . Let the normal at the point j be $\hat{\mathbf{n}}_j$.
2. If $\mathbf{x}_{ij} \cdot \hat{\mathbf{n}}_j > 0$ then the particle i is outside, otherwise it is inside.
3. We perform steps 1 – 2 for all particles near the boundary surface. For particles that are not near the boundary surface, we check the neighbors of the given particle. If a neighbor of a particle is an exterior particle, then it is an exterior particle; otherwise, it is an interior particle. This step works because all particles near a boundary are already marked.

This process provides a simple method of identification of the interior and exterior particles. We note that we do this after the packing procedure is complete. This procedure takes much less computational time compared to the packing procedure.

Implementation

We implement the algorithm using the open-source package PySPH (Ramachandran et al., 2021). In algorithm 5, we show the pseudo-code of the proposed Hybrid method. The

nearest neighbor particle search (NNPS) algorithm implemented in PySPH is not a part of the current algorithm.

The algorithm first reads the input using **ReadInput** and initialize the particles in **CreateParticles**. **SetConstantAndTimeStep** sets the constants and time step. The first iteration starts with **UpdateNeighbors**, where we create the neighbor list for every particle. Next, we perform the acceleration computation in **ComputeAccelerations** and integration in **IntegrateParticles**. **ProjectParticles** converts the nearest free particles and projects them to the surface. This procedure updates the **empty_count** variable. The iterations continue until the convergence condition is true in **CheckConvergence**. Finally, **SeparateParticles** separates the particles into internal and external particles.

Algorithm 5: Hybrid particle packing algorithm.

```

Result: Coordinates of solids and fluids

ReadInput();
CreateParticles();
SetConstantAndTimeStep();
p_freq = 50;
iteration = 0;
empty_count = 0;
converged = False;
while not converged do
    UpdateNeighbors();
    ComputeAccelerations();
    IntegrateParticles();
    if iteration % p_freq == 0 then
        empty_count = ProjectParticles(empty_count);
        if empty_count > 4 then
            converged = CheckConvergence();
        end
    end
    iteration++;
end
SeparateParticles();

```

Obtaining faster convergence

The algorithm discussed above is the basic form, which adds particles slowly to the boundary. In case when the boundary surface is smooth and does not have sharp changes, we can use the following approaches to speed up the packing process:

- Using surface point prediction: When the boundary is smooth, one may project $0.9N_s$ immediately at the start. This bulk projection perturbs the distribution of the particles significantly. If required, we project more particles to the boundary while settling the system to an equilibrium.
- Filter layers near the boundary surface: Conforming particles to the boundary surface requires that we move only some of the particles. Thus, one could filter the free particles near the boundary surface and freeze the other particles.
- Reduce projection frequency: One can reduce it slowly once the initial projection is complete.
- Reduce the convergence tolerance: One can potentially reduce the tolerance to increase the performance of the algorithm, although this would reduce the quality of the distribution of particles.

Doing these can potentially reduce the computations by up to a factor of two. However, we have not performed any of these in the results presented here.

We implement the proposed algorithms in the open-source SPH framework, PySPH (Ramachandran et al., 2021). The current implementation is open source and freely available at https://gitlab.com/pypr/sph_geom. All the results shown in the next section are fully reproducible using an automation framework *automan* (Ramachandran, 2018).

4.2 Comparison of particle packing algorithms

In this section, we compare all the particle packing algorithms discussed in this chapter. We first compare the algorithms for a circular cylinder and a Z-shaped wall in two dimensions. The Standard method is limited to two-dimensional domains, and so in three-dimensions, we compare only the Coupled and the Hybrid method for an ellipsoid. We also show the particle distribution for a symmetric airfoil and an arbitrarily shaped geometry using the Hybrid method at different resolutions. In the end, we demonstrate the Hybrid algorithm for a complex-shaped Stanford bunny. All the dimensions mentioned in this section are in SI units unless stated otherwise.

Circular cylinder

The flow of an incompressible fluid past a cylinder is a well-known benchmark problem. In order to obtain good results, it is important to remove the effect of any surface irregularities due to the underlying method of geometry creation. In this test case, we compare the circular cylinder constructed using all the approaches discussed in the previous sections. We consider a cylinder of diameter $D = 2m$. In fig. 4.11, we show the geometry with particle spacing $\Delta x = 0.1$ constructed using different methods. It is clear that the Hybrid method produces a uniform particle distribution. In the case of the Coupled method, we observe a large number of particles near the wall surface. The Standard method seems to have uniform particles owing to its construction.

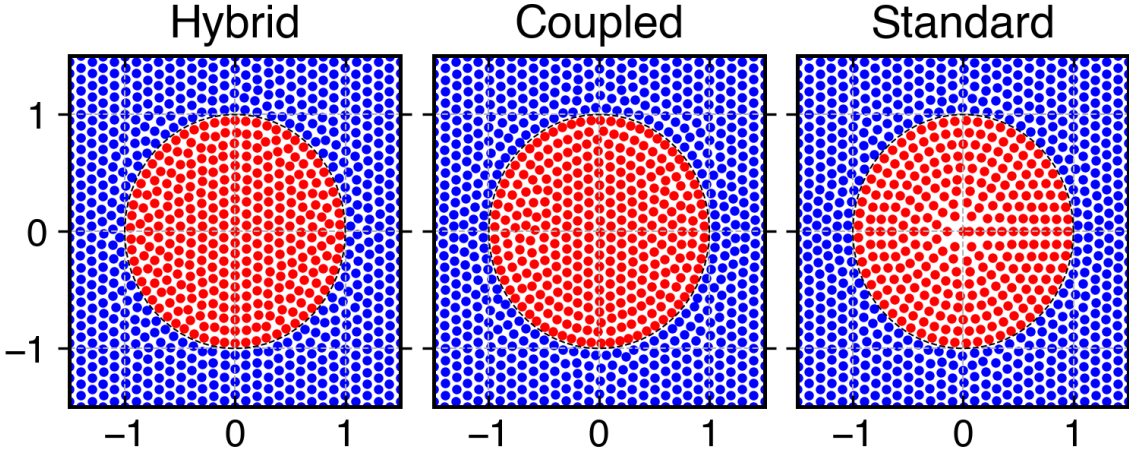


Figure 4.11 : Solid (red) and fluid (blue) particles for a circular cylinder for $\Delta x = 0.1$.

In order to investigate this further, we plot the particle density distribution as shown in fig. 4.12. We obtain the density distribution using eq. (4.1). Clearly, the Coupled method shows high density near the surface, and the Standard method shows a low density on the solid since we assume Δx particle distance over a curved surface. The total density variation is 8%, 16%, and 2% for the Standard, Coupled, and Hybrid methods, respectively. Thus, it is clear that the proposed Hybrid method shows excellent distribution with a maximum variation of 2%.

We study the improvement quantitatively by interpolating a C_∞ function over the packed particles given by

$$f(x, y, z) = \sin(x^2 + y^2 + z^2). \quad (4.24)$$

We approximate the function and its derivative using eq. (1.15) and eq. (1.16), respectively. We evaluate the L_1 error in the approximation using

$$L_1(f - f_o) = \frac{\sum_j |f(\mathbf{x}_j) - f_o(\mathbf{x}_j)|}{N}, \quad (4.25)$$

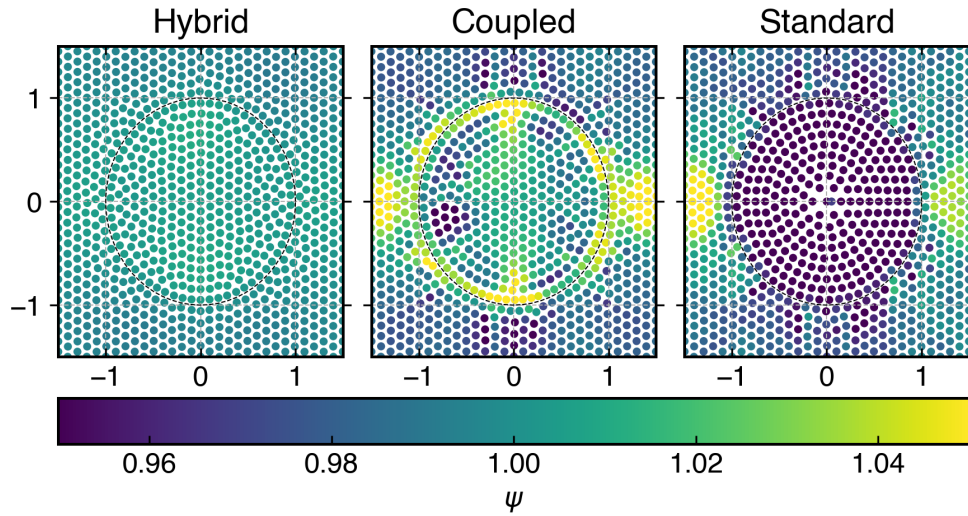


Figure 4.12 : Particle density distribution of the packed particles for the circular cylinder geometry for $\Delta x = 0.1$.

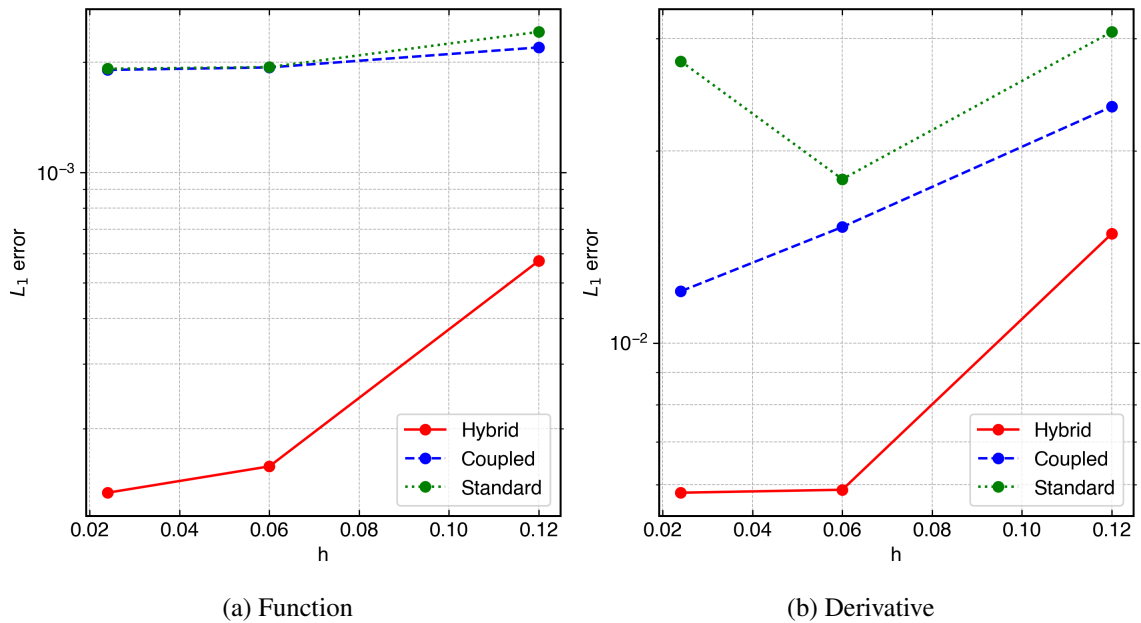


Figure 4.13 : L_1 error for SPH approximation of function and its derivative for the circular cylinder geometry.

where f_o is the SPH function approximation on a regular mesh of points, and N is the total number of particles in the domain. The value of the function is set as per the position of each particle as $f(\mathbf{x}_i)$. This is then interpolated onto a regular mesh using eq. (1.15) for function and eq. (1.16) for its gradient. We do the same for the regular points themselves

to obtain the reference f_o value at each point. We vary the value of h in order to get convergence. We choose a value of $h_{\Delta x} = 1.0$ for $\Delta x = 0.1$ and vary linearly to $h_{\Delta x} = 1.5$ for $\Delta x = 0.02$. We use the quintic spline kernel for the interpolation. When comparing the derivatives, we consider only the x derivative.

In fig. 4.13a and fig. 4.13b, we show L_1 error in function and its derivative approximation. In these plots, we do not evaluate the errors near the center of the cylinder since they do not affect the flow, and also, the Standard method performs poorly in this region. Hence, the L_1 norm is evaluated only over the points where $r > 0.45 D$ as this allows for a fair comparison. It is clear that the Hybrid method shows a significant order of magnitude improvement compared to both Coupled and Standard methods. The Coupled method is slightly better than the Standard method. The Standard method shows large errors due to the way in which we represent the surface.

Zig-Zag Wall

The zig-zag wall is one of the test cases proposed by Marrone et al. (2011) used to demonstrate the δ -SPH method. They employ the Standard packing algorithm in order to generate a solid body and pack the fluid particles around. In this test case, we pack particles using all the algorithms and compare the density and function approximations as done earlier. The zig-zag wall is an excellent test case for packing since it has both concave and convex sharp edges. In order to generate a solid using the Standard method, we move the corner nodes along the angle bisector and generate uniform points using these points as endpoints. In the other two algorithms, we refer to these sharp points as corner nodes and employ the method discussed in section 4.1.3 to automatically restrict the motion of these points.

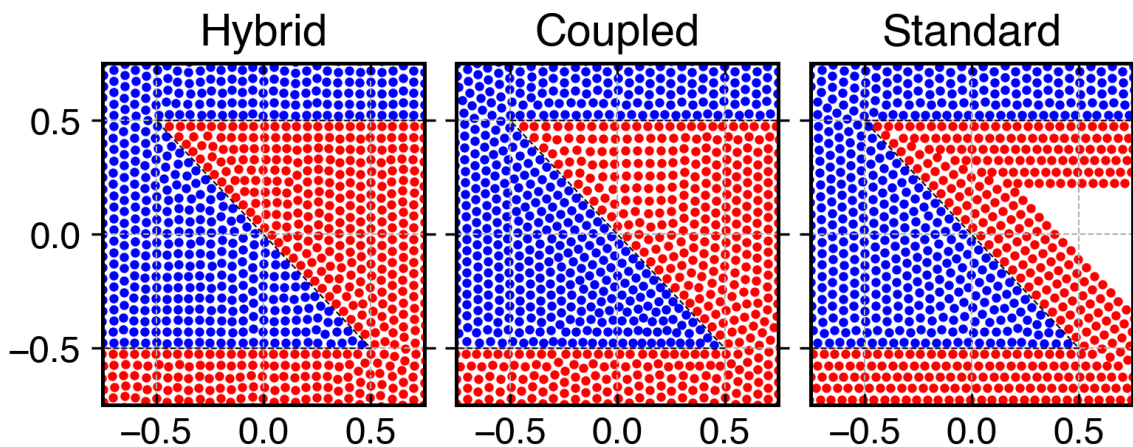


Figure 4.14 : Solids (red) and fluids (blue) for the zig-zag wall for $\Delta x = 0.05$.

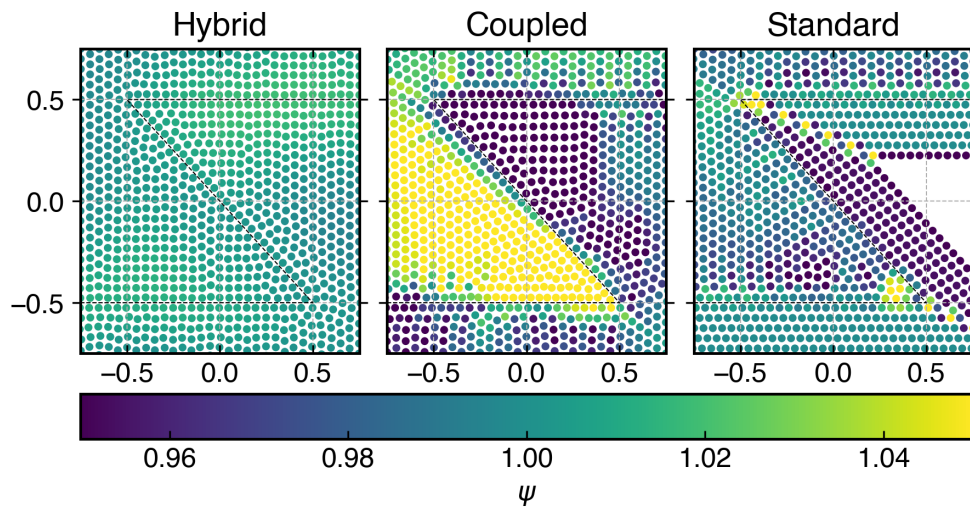


Figure 4.15 : Particle density distribution for the zig-zag wall for $\Delta x = 0.05$.

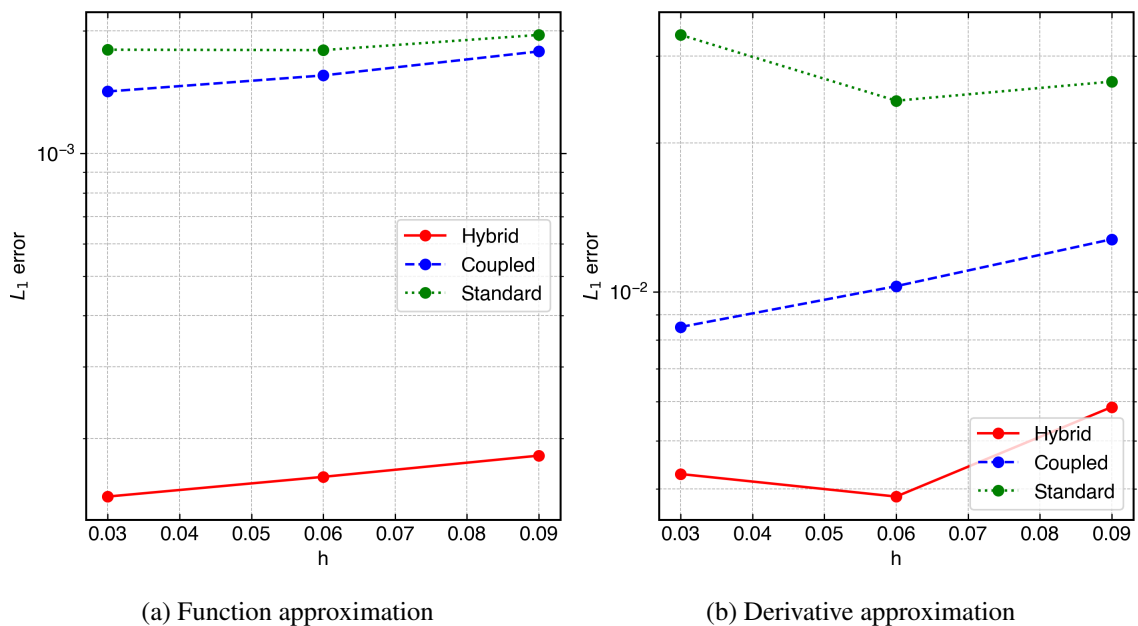


Figure 4.16 : L_1 error for SPH approximation of function and its derivative for the zig-zag wall.

In fig. 4.14, we show the solid and fluid particles packed with $\Delta x = 0.05$ using different algorithms. It is difficult to conclude which of these is better. However, on looking at the density distribution computed using the summation density as shown in fig. 4.15, the proposed method has much less deviation from the desired density. In the case of the Coupled method, we observe higher density at the concave corner. This increase oc-

curs since the particles from both sides push towards the interface at the first pass of the Coupled algorithm as discussed in section 4.1.2. The Standard method shows an uneven variation of density near the sharp edges, which is not desirable. The total density variation is 10%, 30%, and 3% for Standard, Coupled, and Hybrid methods, respectively. It clearly shows that the Hybrid method shows very small density variations compared to other methods.

We perform a similar analysis over the zig-zag wall as done in the case of the cylinder. In order to remove the effect of the interior of the solid in the standard case, we compute the errors only up to a distance of Δx inside the wall. In fig. 4.16a and fig. 4.16b, we plot the L_1 norm for the error in function and its derivative SPH approximation, respectively. Clearly, the proposed method performs very well compared to the other methods.

Packing at different resolutions

In this example, we apply the Hybrid algorithm to an arbitrarily-shaped body. We show the packing at different particle spacings in fig. 4.17. The particle spacings chosen are 0.05, 0.075, and 0.1. We place the particles at $\Delta x/2$ distance away from the boundary. Clearly, the density distribution is close to the desired value of 1.0. The particles conform to the body surface and have a total variation of density of 2.5%, 2%, and 3% for particle spacing 0.1, 0.075, and 0.05, respectively. It shows that the proposed algorithm is applicable to complex two-dimensional geometries.

Effect of convergence tolerance on the quality

This test case shows that the proposed algorithm can achieve a high-quality particle distribution for a given tolerance. We consider a symmetric airfoil, NACA0015. We choose the endpoint at the trailing edge as a corner node. We perform particle packing using the proposed Hybrid method for the spacing $\Delta x = 0.02$.

Tolerance	Iterations	$L_\infty(\psi_i)$
2.5e-05	19641	0.011
5.0e-05	6291	0.019
1.0e-04	1191	0.028

Table 4.1 : The table shows the effect of the change in tolerance for convergence on the number of iterations and the L_∞ error in the density.

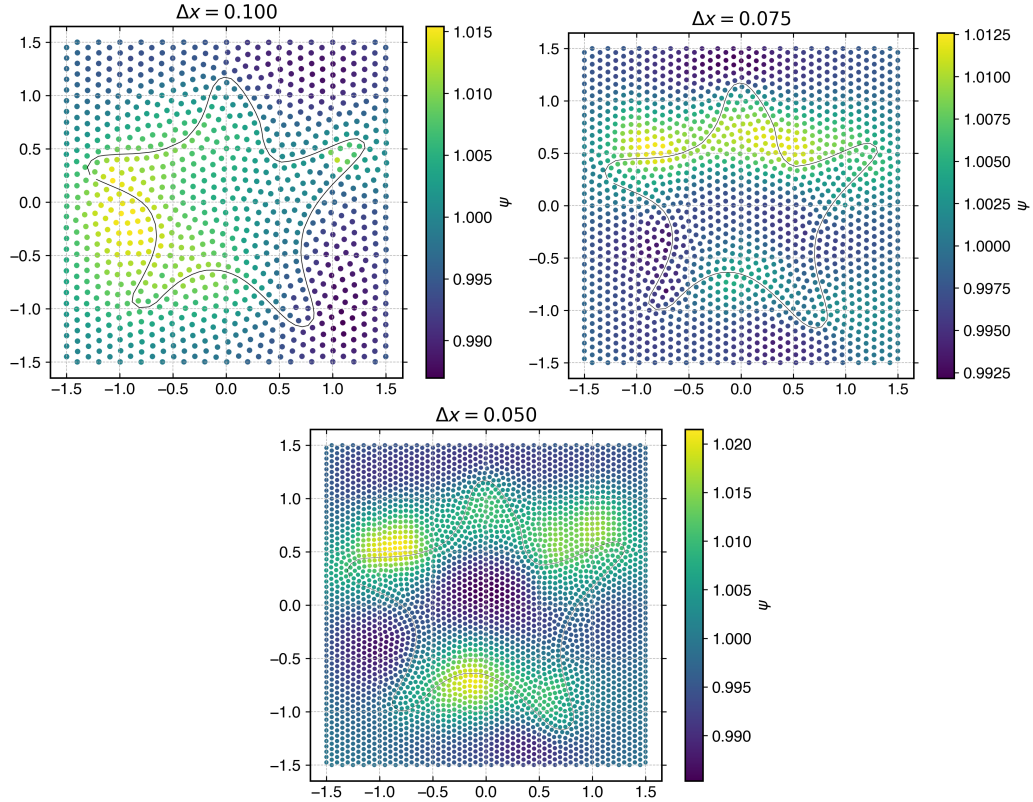


Figure 4.17 : Particle density distribution at different resolutions for an arbitrarily-shaped object.

In table 4.1, we tabulate the number of iterations required for convergence and the L_∞ error in the density for different values of the tolerance required for convergence. It is evident from the table that the L_∞ error decreases with the decrease in the tolerance; however, the number of iterations required increases significantly. In the fig. 4.18, we show the high-quality particle distribution achieved using a lower value of tolerance, $\epsilon = 2.5 \times 10^{-5}$. It shows that the proposed method can achieve a high-quality particle distribution provided we use a sufficiently low tolerance.

Particle Packing in 3D

One of the advantages of the proposed algorithm is that it can be easily extended to a three-dimensional object, unlike the Standard method. In order to compare the packing in 3D, we pack particles for a simple ellipsoid. The ellipsoid has semi-major axis dimensions, $a = 1.0$, $b = 0.5$ and $c = 0.75$ along x , y and z axis, respectively.

In fig. 4.19a and fig. 4.19b, we show the packed particles over the surface of the sphere for the Hybrid and Coupled methods, respectively. The color of the particles shows the density distribution. In order to show that the particles conform to the surface, we pull the surface along the normal by $\Delta x/2$. It is clear that the Hybrid method attains a good

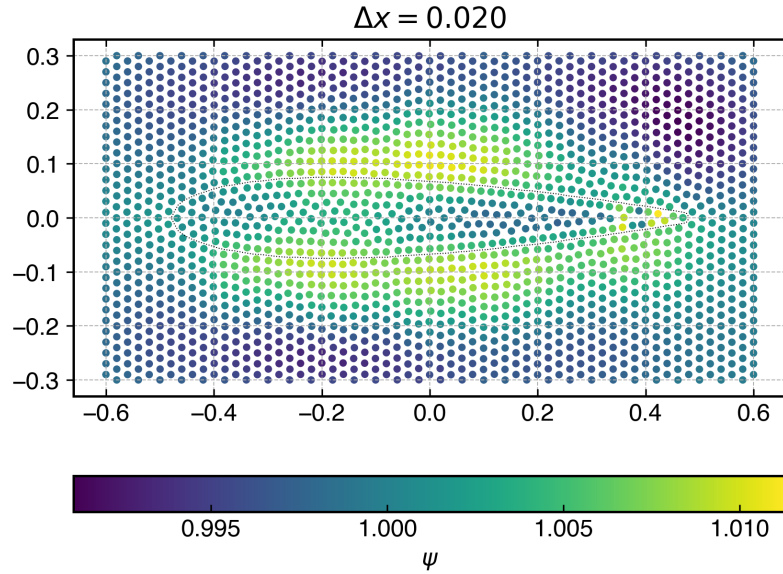


Figure 4.18 : The density distribution for a symmetric airfoil with particle spacing $\Delta x = 0.02$ and convergence tolerance $\epsilon = 2.5 \times 10^{-5}$.

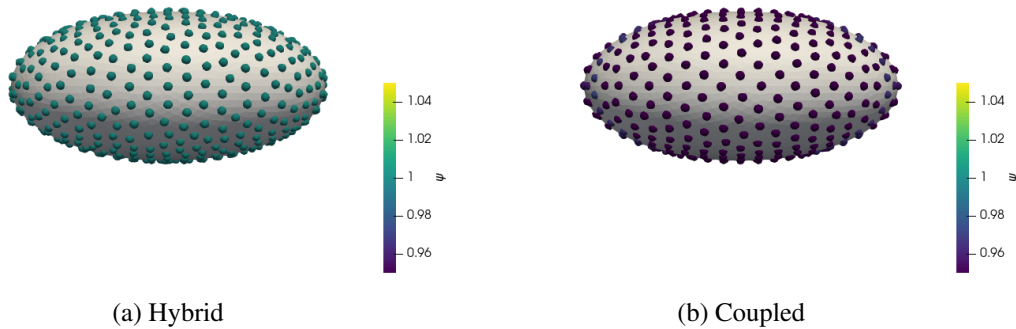


Figure 4.19 : Particle density distribution on the surface of the ellipsoid for $\Delta x = 0.1$.

distribution of particles resulting in a density distribution very close to $\psi_o = 1.0$. The particle distribution using the Coupled method has a density near the lower range of the scale. It is due to the fact that, unlike the Hybrid method, the Coupled method does not project the required number of particles on the surface.

In order to perform a quantitative analysis, we adopt the comparison of the function and derivative approximation used earlier. In fig. 4.20a and fig. 4.20b, we plot L_1 error in SPH approximation of function and its derivative, respectively. As can be seen, the proposed method produces much lower errors at lower resolutions.

In order to show the capability of the algorithm, we also apply the proposed algorithm to pack particles around the Stanford bunny geometry, as done by Jiang et al. (2015). The geometry surface must have outward normals, and we correct the mesh using

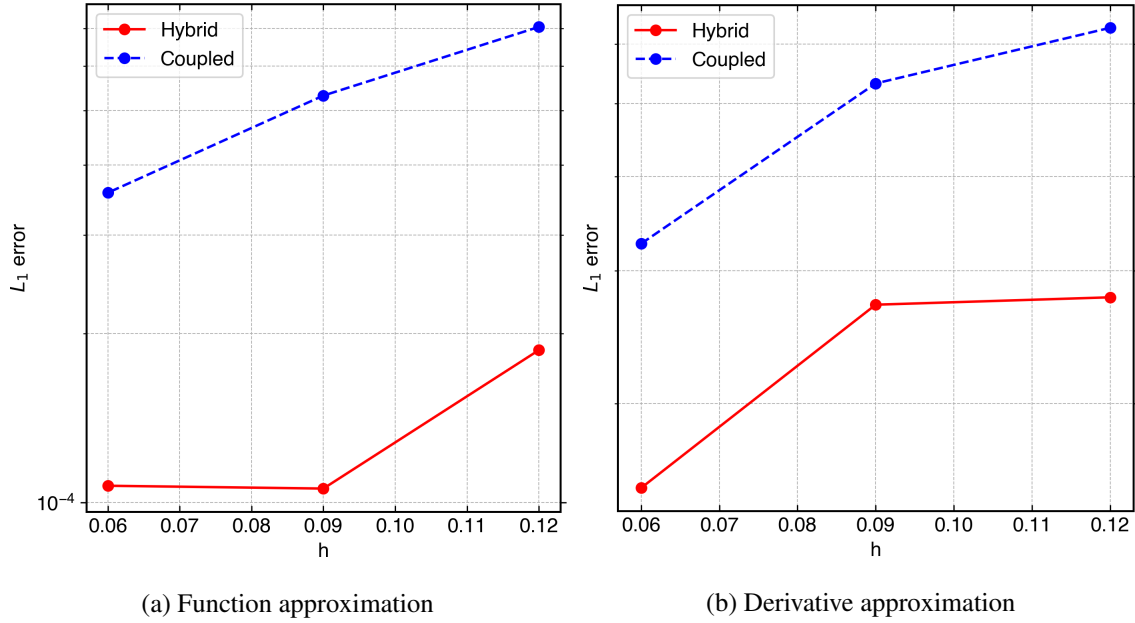


Figure 4.20 : L_1 error for SPH approximation of function and its derivative for the ellipsoid.

a mesh manipulation tool. We choose a particle spacing of 0.02. In this case, the intention is to show how well we capture the geometry so we do not shift the surface inside. In fig. 4.19b, we show the particle distribution over the surface of the bunny. The results show the applicability of the proposed algorithm to arbitrary-shaped 3D objects. After pre-processing, we can place the 3D object anywhere in the domain with the surrounding particles.

4.3 Summary

In this chapter, we propose an improved particle packing algorithm to simulate flows involving complex geometries in two and three dimensions. We implement and compare three different methods for packing particles around an arbitrary-shaped object, namely, (i) the Standard method proposed by Colagrossi et al. (2012) along with the solid object construction proposed by Marrone et al. (2011), (ii) the Coupled method, a modified version of that proposed by Jiang et al. (2015), handles both the interior and exterior of the body, and (iii) a new method that combines the best features of these methods. The proposed method provides a uniform density distribution as a result of evenly distributed particles. The method applies to both 2D and 3D domains. Unlike the Coupled method, we do not require estimation of particles inside and outside. We show several benchmark cases which highlight the accuracy of the proposed algorithm in two and three dimensions.

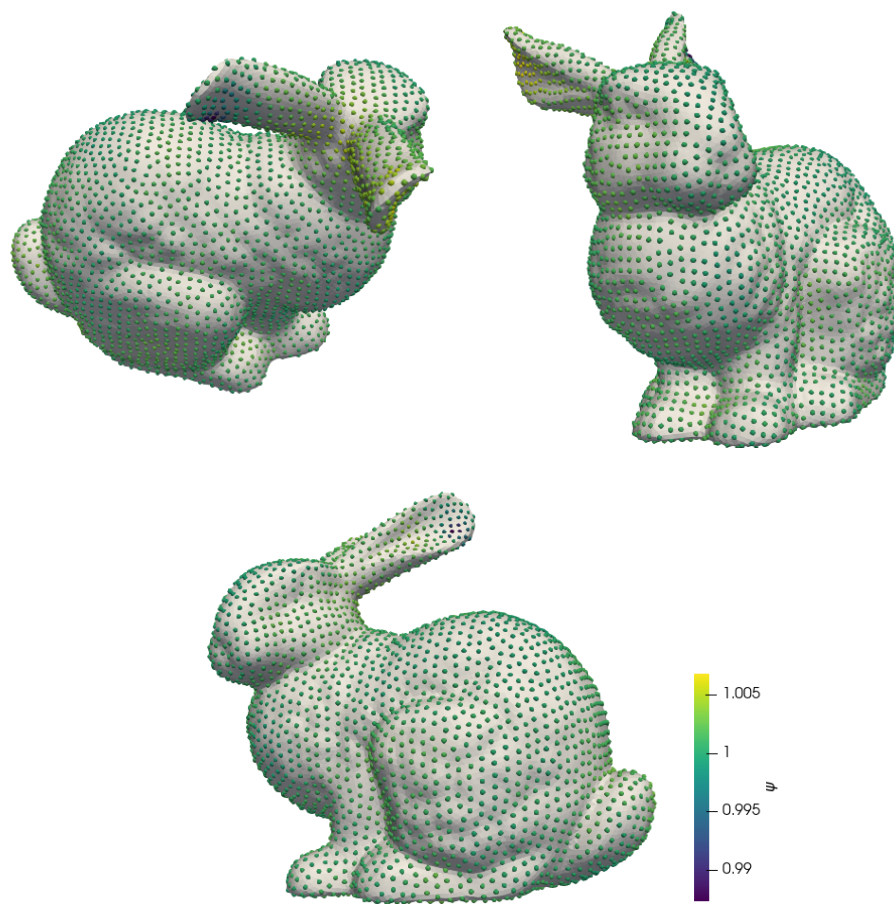


Figure 4.21 : Different views showing particles (colored with particle density values) over the Stanford bunny.

In the next chapter, we discuss various methods to implement Neumann pressure, slip, no-slip, inflow, and outflow boundary conditions in SPH. We use the proposed packing method to construct boundary conforming solid and fluid particles for the test cases for verification of these boundary condition implementations.

Chapter 5

Boundary condition implementations in SPH

In SPH, the boundaries are realized using particles that may or may not interact with the fluid particles. In this chapter, we discuss various methods in the SPH literature used to implement boundary conditions (BCs). We classify these implementations based on the requirement of the secondary particle arrays. In SPH, two types of particles are used to implement boundary conditions, viz. ghost and virtual particles. The ghost particles carry the extrapolated properties from the fluid and influence the fluid particles. Whereas the virtual particles are used to evaluate some intermediate value of a property from fluid and do not affect the actual flow. In the case of the solid boundary, researchers have proposed different methods to either ensure force balance or accurate function extrapolation. However, in the case of open boundary, along with accurate function extrapolation, non-reflection of the pressure waves is an additional requirement. We propose a novel implementation that ensures both conditions. Furthermore, we demonstrate that the traditional test cases to identify non-reflecting nature need to be revised. Therefore, we propose a suite of new test cases that tests specific feature of a non-reflecting boundary condition (NRBC) implementation. In the next section, we discuss various solid boundary implementations followed by open boundary conditions.

5.1 Solid boundary conditions

Many authors (Ferrand et al., 2013; Hashemi et al., 2012; Marongiu et al., 2007) have proposed different approaches where a single layer of ghost particles is used. In order to assess the effect of the number of layers on the accuracy of the second-order accurate gradient and Laplacian discretization, we perform a simple numerical test. In this test, we

consider a finite 2D domain of size $1m \times 1m$. We discretize the domain using particles at different resolutions and initialize various properties using

$$\begin{aligned} u(x, y) &= \sin(4\pi(x + y)), \\ v(x, y) &= \cos(4\pi(x + y)), \\ p(x, y) &= \sin(4\pi x) + \sin(4\pi y). \end{aligned} \quad (5.1)$$

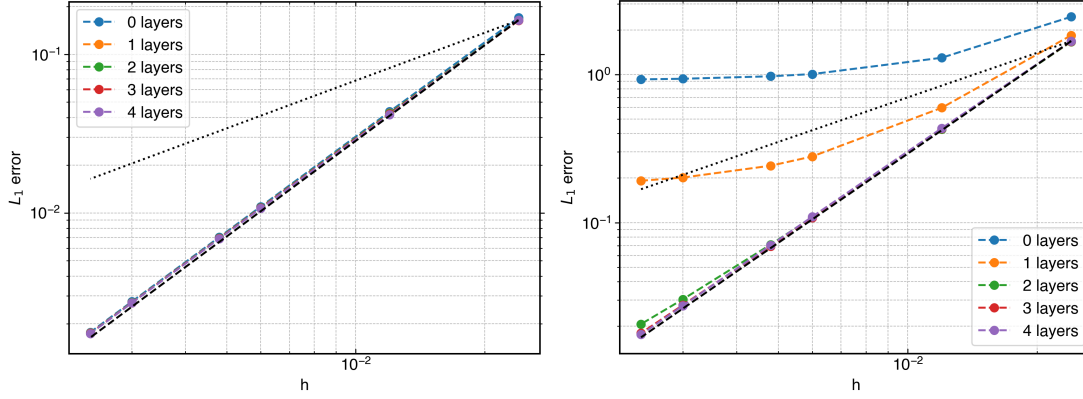


Figure 5.1 : L_1 error in pressure gradient (left) and Laplacian (right) approximation with change in the skipped number of layers.

We compute the SPH approximation of the pressure gradient and Laplacian of velocity on the particles using the discretization used in the L-IPST-C scheme. Since the particles near the boundary do not have complete support, they will have higher errors compared to the inner particles. We compute the L_1 error in the approximation using

$$L_1(N) = \frac{\sum_j^N |f(\mathbf{x}_j) - f_o(\mathbf{x}_j)|}{N}, \quad (5.2)$$

where $f(\mathbf{x}_i)$ is the computed value and $f_o(\mathbf{x}_i)$ is the actual value, N are the number of particles in the domain $(1 - 2l\Delta x) \times (1 - 2l\Delta x)$, where l is the number of layers skipped from each side. In fig. 5.1, we plot the L_1 error for the particles skipping l -layers of particles from the outermost boundary for pressure gradient and Laplacian approximations. We note that both discretizations used are second-order accurate in a periodic domain (see section 2.2). However, we observe that the pressure gradient is second-order accurate even when zero layers of particles are surrounding the domain of interest. Whereas for the Laplacian approximation, we require at least 2 layers of particles. Since all the second-order accurate formulations for Laplacian approximation require a gradient on all the neighboring particles (see section 2.2.3) therefore the requirement for the number of neighboring particles is double to that in the case of the gradient approximation for the second-order accurate convergence.

This numerical test shows that the boundary implementation, which uses a single layer of particles is bound to have an error in Laplacian approximation resulting in an inaccurate solution, irrespective of having an accurate boundary condition implementation. Furthermore, since all the second-order viscosity formulations require the evaluation of the gradient on the boundary particle as discussed in section 2.2.3, we do not pursue the implementation of boundaries based on the local point symmetry method in Ferrari et al. (2009) and Fourtakas et al. (2015; 2019) for the no-slip boundary condition. The various solid boundary implementations considered are as follows:

I. Using a single layer of ghost particles on the boundary surface

A. With virtual particles

In these methods, only one layer of particles is used on the boundary surface. Marongiu et al. (2007) proposed a characteristics-based evolution equation for density update at these boundary particles, given by

$$\frac{d\rho}{dt} = c_o \frac{\partial \rho}{\partial \hat{\mathbf{n}}} - \rho \frac{\partial u_n}{\partial \hat{\mathbf{n}}} - \frac{\rho \mathbf{g} \cdot \hat{\mathbf{n}}}{c_o}, \quad (5.3)$$

where $\hat{\mathbf{n}}$ is the normal of the boundary surface pointing into the fluid, $\frac{\partial(\cdot)}{\partial \hat{\mathbf{n}}}$ is the directional derivative along the normal, and $u_n = \mathbf{u} \cdot \hat{\mathbf{n}}$. In order to evaluate the gradient at the boundary point five-point finite difference approximation is used. These five points are generated along the normal of the boundary particle at a spacing equal to the average particle spacing represented by black points in fig. 5.2 for a single particle. The values of the properties at these black points are evaluated using Shepard interpolation (Shepard, 1968).

B. Without virtual particles

Hashemi et al. (2012) proposed methods to implement pressure and no-slip boundary conditions using one layer of boundary particles on the surface. However, they do not use any extra set of elements to derive the values on the boundary elements denoted by red particles in the fig. 5.2. In order to satisfy the no-slip boundary condition the velocity of the red particles is kept the same as the velocity of the solid boundary. Whereas for pressure boundary implementation, momentum balance is performed along the normal of the boundary surface, given by

$$\frac{\nabla p}{\rho} \cdot \hat{\mathbf{n}} = -\frac{d\mathbf{u}}{dt} \cdot \hat{\mathbf{n}} + \frac{\mu \nabla^2 \mathbf{u}}{\rho} \cdot \hat{\mathbf{n}} + \mathbf{g} \cdot \hat{\mathbf{n}}, \quad (5.4)$$

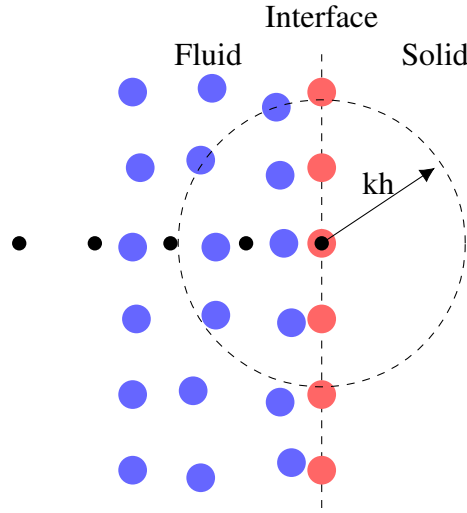


Figure 5.2 : Arrangement of ghost and virtual particles for a given fluid particle denoted by blue circles. The dashed line is the boundary surface, red particles are ghost (solid) particles. The black dots are generated to evaluate the gradient using finite differences at the red boundary points.

where \mathbf{n} is the normal of the boundary surface pointing into the fluid. Equation (5.4) is discretized using the second-order consistent approximation to obtain the pressure at i^{th} solid boundary particles, given by

$$p_i = \frac{\left(\frac{p_j}{\rho_i} \nabla \tilde{W}_{ij} \omega_j\right) \cdot \hat{\mathbf{n}}_i - \left\langle -\frac{d\mathbf{u}}{dt} \cdot \hat{\mathbf{n}} + \frac{\mu \nabla^2 \mathbf{u}}{\rho} \cdot \hat{\mathbf{n}} + \mathbf{g} \cdot \hat{\mathbf{n}} \right\rangle_i}{\left(\frac{1}{\rho_i} \nabla \tilde{W}_{ij} \omega_j\right) \cdot \hat{\mathbf{n}}_i}, \quad (5.5)$$

where $\nabla \tilde{W}_{ij}$ is the corrected kernel gradient.

II. Using multiple layers of ghost particles outside boundary surface

A. With virtual particles

Marrone et al. (2011) proposed a method where fixed virtual particles are generated by reflecting the ghost particles about the interface. The created particles are illustrated in the fig. 5.3, where red crosses are the virtual particles and red particles on the right represent the ghost particles. The properties on the ghost particles are set as $\rho_g = \rho_v$, $p_g = p_v$, and the velocity is set according to the slip or no-slip condition required, where $*_g$ represent the property value on ghost particle and $*_v$ represent the property value on the corresponding virtual particle. In the case of the slip boundary, the velocity normal to the wall is reversed, whereas in the case of the no-slip boundary, the velocity is set negative of the value on the corresponding virtual particle. The properties of the virtual particles

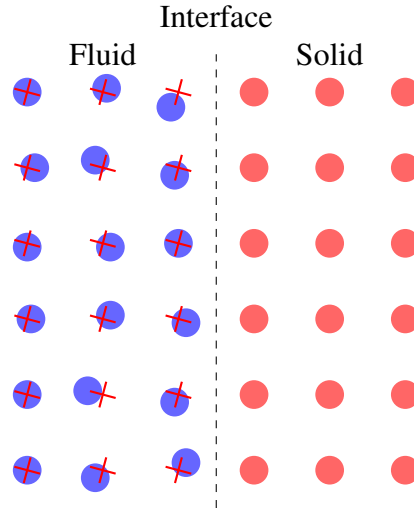


Figure 5.3 : Arrangement of ghost and virtual particle for given blue fluid particles. The dashed line shows the solid boundary. The red circles on the right are the ghost particles, and the red crosses are created by reflecting the ghost about the interface.

are evaluated using

$$\phi_i = \sum_j \phi_j \tilde{W}_{ij} \omega_j, \quad (5.6)$$

where ϕ_* is the desired property, ω_j is the truncated volume of the fluid particles and \tilde{W}_{ij} is the kernel corrected using the method by Liu et al. (2006). The sum j is taken over all the fluid particles in support of the kernel,

B. Without virtual particles

In the SPH literature, most of the methods for boundary condition implementation use multiple layers of ghost particle such that the kernel has full support. Takeda et al. (1994) proposed to extrapolate the properties using a linear interpolation depending upon the distance of the ghost particle from the nearest fluid particle shown in fig. 5.4. The extrapolated velocity is given by

$$\mathbf{u}_j = -\mathbf{u}_i \frac{r_j - r_o}{r_o - r_i}, \quad (5.7)$$

where $r_j - r_o$ and $r_o - r_i$ are the distances along the normal from the boundary for j^{th} ghost and i^{th} fluid particle, respectively. r_o is the location of the boundary surface intersection the line joining the i^{th} and j^{th} particle. We select the nearest fluid particle as the i^{th} for a corresponding j^{th} ghost particle along the normal of the boundary.

Randles et al. (1996) proposed a boundary condition for pressure. The prescribed pressure value p_{bc} is assigned to the ghost particles in red, and the values on the light

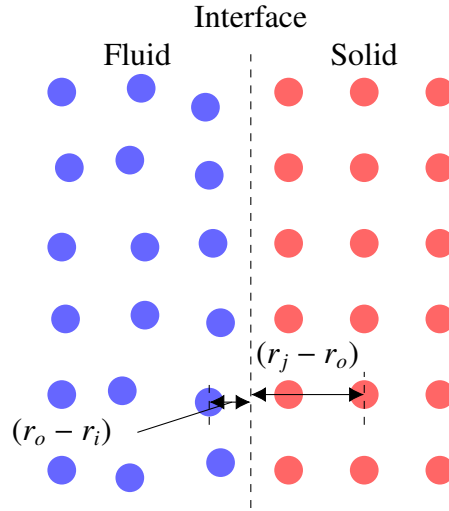


Figure 5.4 : Arrangement of particles for the method proposed by Takeda et al. (1994).

The ghost particles are shown in red, and the fluid particles are shown in blue color. The value of the red particles are interpolated using the nearest fluid particle along the normal of the boundary surface.

blue fluid particles as shown in fig. 5.5 are set such that the desired boundary condition is satisfied. The value of light blue fluid particles is given by

$$p_i = p_{bc} + \frac{\sum_{j \in I} (p_j - p_{bc}) W_{ij} \omega_j}{1 - \sum_{j \in B} W_{ij} \omega_j}, \quad (5.8)$$

where I is the set of blue particles, B is the set of red particle, and pressure is evaluated at each i^{th} light blue particle shown in fig. 5.5.

Adami et al. (2012) proposed the method where Shepard interpolation is employed to extrapolate properties from fluid to ghost particles. The property at a ghost particle is given by

$$\phi_i = \frac{\sum_j \phi_j W_{ij}}{\sum_j W_{ij}}, \quad (5.9)$$

where sum j is over all the fluid particles in support of the kernel as shown by red dashed circle in fig. 5.5, at the i^{th} ghost particle. Furthermore, Esmaili Sikarudi et al. (2016) proposed to perform a first-order accurate extrapolation to evaluate the properties of ghost particles.

Colagrossi et al. (2003) proposed to mirror the fluid particles near the solid interface, about the interface to generate ghost particles as shown in fig. 5.6. These ghost particles carry the velocity of the opposite sign to implement no penetration. The value of pressure and density is kept the same. In order to implement this method, we create new particles for solids from the fluid particles before each time step.

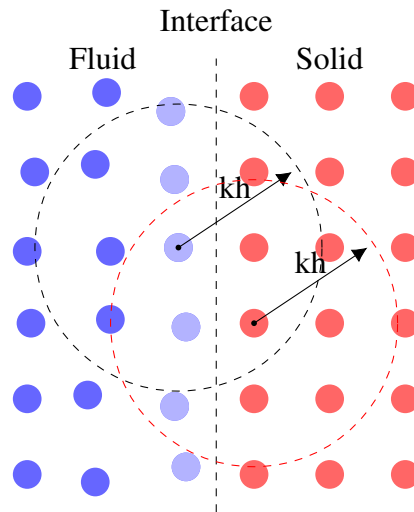


Figure 5.5 : Arrangement of particles for the method proposed by Randles et al. (1996) and Adami et al. (2012) for the fluid particles in blue. The boundary interface is shown by the dashed line. The red circles on the left represent the ghost solid particles. The property of the light blue fluid particles are manipulated to satisfy boundary conditions.

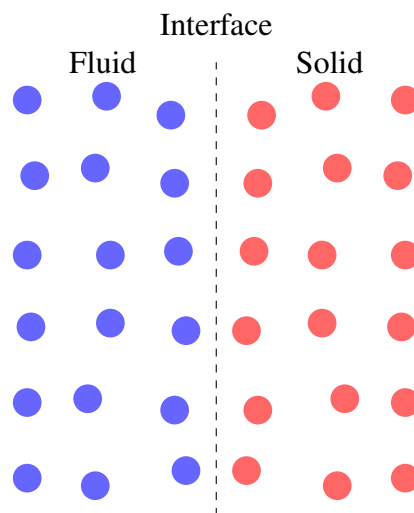


Figure 5.6 : Arrangement of particles for the method proposed by Colagrossi et al. (2003) for the fluid particles in blue, and boundary represented by the dashed line. The ghost particles in red represent the solid created by the mirror reflection of fluid particles about the interface.

5.2 Open boundary conditions

Open boundary conditions are required to simulate a wind-tunnel kind of simulation in SPH. It consists of an inlet from where the particles are added to the domain and an outlet from where the particles exit the domain as shown in fig. 5.7. The particles added to the domain should have prescribed inlet velocity and should not introduce any artifacts in the flow, whereas the outlet should remove particles from the flow without affecting the flow. Since we use a weakly-compressible scheme, pressure waves are inevitable. Therefore, the inlet/outlet must be non-reflecting along with adherence to the boundary condition. In fig. 5.7, we show a schematic arrangement of the inlet, fluid, and outlet domains. The dashed lines between different regions are the boundary surfaces. The inlet/fluid particles are converted to fluid/outlet particles once they cross the boundary surface. Many authors (Federico et al., 2012; Lastiwka et al., 2009; Tafuni et al., 2018) have proposed different methods to implement inlet/outlet boundary conditions in SPH. Open boundary condition implementations considered in this work are as follows:

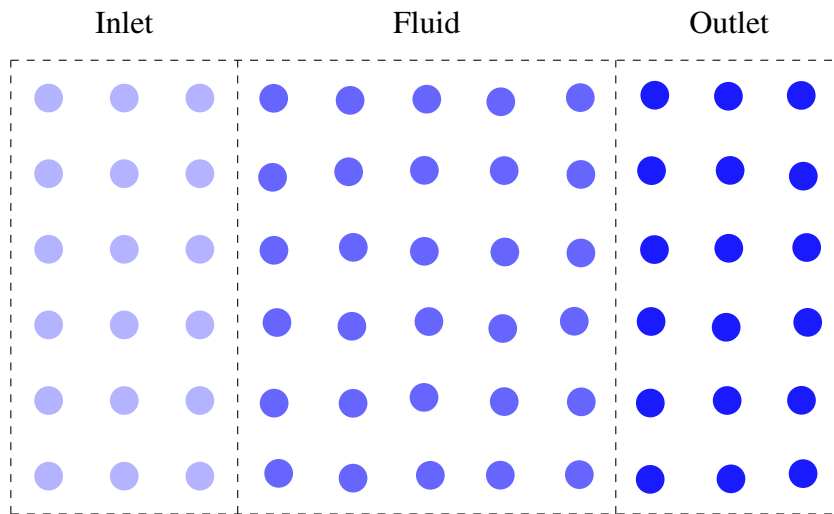


Figure 5.7 : Schematic of the arrangement of fluid with the inlet and outlet particles.

The top and bottom are supported by solid particles not shown in the figure.

I. Do-nothing

Federico et al. (2012) proposed the method to implement outlet boundary where the properties of the fluid particles crossing the fluid-outlet interface are fixed in time. Therefore, the particle, after crossing the interface, advects with a constant velocity.

We propose a subtle modification to the standard do-nothing method described above. Unlike the standard do-nothing where the outlet moves with a velocity with which

it left the fluid domain, we propose to extrapolate the transport velocity of the fluid to determine the transport velocity of the outlet particles. Thus the advection is given by

$$x_o^n = x_o^{n-1} + \tilde{u}^n \Delta t, \quad (5.10)$$

where \tilde{u}^n is the Shepard extrapolated fluid velocity at timestep n given as

$$\tilde{u}_i = \frac{\sum_j \tilde{u}_j W_{ij}}{\sum_j W_{ij}}, \quad (5.11)$$

where the sum is taken over all the fluid particles j in the neighborhood of i_{th} outlet particle. It must be noted that the advection velocities are only used to advect the particles and the actual velocity of the outlet particles remain the ones frozen when the fluid particle is converted to the outlet particle. We refer to this method as modified do-nothing.

II. Mirror

Tafuni et al. (2018) proposed a method applicable to both inlets and outlets. The properties from the fluid are stored on the virtual particles generated by reflecting the inlet/outlet particles about the interface, as shown in fig. 5.8. The desired property and its gradient are evaluated for each mirror particle using a first-order consistent approximation. From the mirror particle, the property of the corresponding outlet/inlet particle is evaluated using

$$\phi_o = \phi_m - \Delta x_{om} \nabla \phi_m, \quad (5.12)$$

where ϕ_m and $\nabla \phi_m$ are the properties of the mirror particle and Δx_{om} is the distance between the outlet and mirror particle. In addition to this method, we test the convergence of the method when the Taylor series correction is not applied such that the eq. (5.12) is simplified to

$$\phi_o = \phi_i. \quad (5.13)$$

We refer to this method as simple-mirror.

III. Hybrid

Lastiwka et al. (2009) proposed a non-reflecting inlet and outlet boundary implementation using the method of characteristics (MOC). The characteristics of the flow are given by

$$\begin{aligned} J_1 &= -c_o^2(\rho - \rho_o) + (p - p_o), \\ J_2 &= \rho c_o(u - u_o) + (p - p_o), \\ J_3 &= -\rho c_o(u - u_o) + (p - p_o), \end{aligned} \quad (5.14)$$

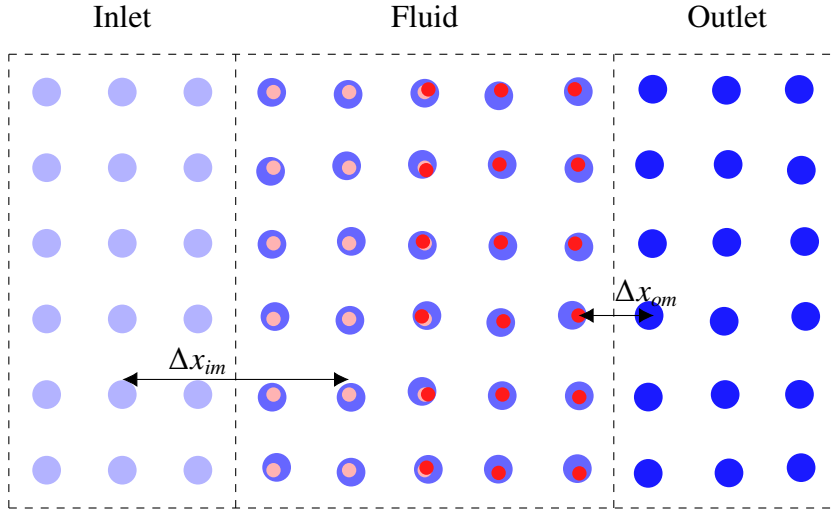


Figure 5.8 : Schematic of the arrangement of fluid with the inlet and outlet particles and their corresponding virtual particles are shown in light red and red, respectively.

where ρ_o , u_o , and p_o are the reference density, velocity normal to the outlet, and pressure. These characteristics are extrapolated to the inlet/outlet particles, and the properties are calculated using the extrapolated characteristics, given by

$$\begin{aligned}\rho &= \rho_o + \frac{1}{c_o^2} \left(-J_1 + \frac{1}{2}J_2 + \frac{1}{2}J_3 \right), \\ u &= u_o + \frac{1}{2\rho c_o} (J_2 - J_3), \\ p &= p_o + \frac{1}{2} (J_2 + J_3).\end{aligned}\tag{5.15}$$

In order to implement the inlet boundary, J_1 and J_2 should be set to zero and J_3 must be extrapolated from the fluid to the inlet particles. On the other hand, for outlet J_1 and J_2 are extrapolated from the fluid to the outlet, and J_3 is set to zero.

Lastiwka et al. (2009) do not specify the method to obtain the reference values in eq. (5.14). Therefore, we describe a hybrid method to implement outlet/inlet boundaries that combines the features of the do-nothing method and the method of characteristics discussed above. At the inlet/outlet, essentially two kinds of fluctuations are encountered, namely spatial variations, which do not change rapidly in time, and variations due to acoustic waves, which travel with the artificial speed of sound. The weakly compressible SPH schemes generate perturbations that travel with the prescribed speed of sound, unlike ISPH schemes which solve for a pressure-Poisson equation. In the case of a do-nothing type of outlet boundary, as described earlier, the particle properties are frozen. As a result, when the acoustic wave arrives at the outlet, its velocity suddenly drops to the particle velocity in the outlet. This causes an increase in the pressure for particles that are near the

outlet. We propose a method to separate the fluid flow properties into acoustic and base flow properties.

A time-averaged property of the flow is given by

$$\phi_{avg} = \frac{\sum_{n=1}^{N_t} \phi_n}{N_t}, \quad (5.16)$$

where ϕ is the fluid property, N_t is the number of time steps used in the averaging. The value of N_t can be estimated by determining the number of time steps the acoustic wave takes to move from one particle to another, given by

$$N_t = \frac{\Delta x}{\Delta t(U + c_o)}. \quad (5.17)$$

In all our test cases, $N_t \approx 4$. Thus, in order to have an optimum mean, we take $N_t = 6$ for all our test cases. Further, in order to detect the acoustic wave, the acoustic intensity is used as a parameter. The time-averaged properties are not changed whenever the acoustic intensity I of the flow is greater than the prescribed value I_o . The acoustic intensity I is given by $p^2/(2\rho c_o)$ (Kinsler et al., 1999). The prescribed value of acoustic intensity I_o can be determined using the inlet velocity, $|\mathbf{u}_i|$ and is given by

$$I_o = \frac{(\frac{1}{2}\rho|\mathbf{u}_i|^2)^2}{2\rho c_o}. \quad (5.18)$$

The difference between the particle property and its time average gives us the acoustic component. The time-averaged part is advected out of the domain using the do-nothing method. Since the acoustic wave travels with the speed of sound, it should be propagated out with the same. We use the method of characteristics described above to propagate these acoustic perturbations into the outlet, where the reference values are the time averages. In our implementation, we keep ρ_o fixed.

When a particle moves from the fluid domain into the outlet, it retains its time average values. The acoustic properties are added to this using Shepard interpolation to the outlet zone as

$$\phi_o = \phi_{ac} + \phi_{avg}, \quad (5.19)$$

where ϕ_{ac} is determined using the extrapolated J_2 as explained above. Since the do-nothing condition is used at the outlet for the time-averaged values, the proposed method cannot simulate incoming flow near the outlet; however, it is suitable for wind tunnel type of flow where the flow always exits the outlet from one side. The particles in the outlet layer are advected using the velocity evaluated with the equation (5.19) (assuming the outlet is perpendicular to the x -axis). The particles are not moved in the transverse direction. We note that the Shepard interpolation of the properties from the fluid will not

always get extrapolated to all the particles in the outlet. These particles are advected with the average of the existing outlet advection velocity. In the inlet, the ϕ_{avg} do not change much, and the acoustic part is simply added as in the case of outlet. In the next section, we propose various test cases to compare the non-reflecting boundary condition (NRBC) implementations discussed above.

5.2.1 Test cases for open BC implementations

In this section, we compare the different methods for the non-reflecting with a variety of test cases. We first simulate the flow past a backward-facing step and a circular cylinder. We employ the EDAC scheme (see appendix A.5.4) to solve the governing equations. We apply the mirror method at the inlet and different NRBC implementations at the outlet. We show that these test cases are not only computationally expensive but also unable to contrast against a method that reflects the pressure waves.

We subsequently propose various test cases that take a small amount of computational effort and highlights specific issues. The new problems are all two-dimensional, and this makes them relatively easy to implement. They include a one-dimensional pulse (in a two-dimensional domain), a two-dimensional pulse, a two-dimensional vortex, and a ramp inlet condition in order to test the typical conditions that outlets encounter. In order to obtain a solution representing an infinite domain for comparison, we simulate the flow in a very long domain. The properties of the fluid are measured at a probe placed inside the domain at a distance d from the inlet. The length of the domain L is chosen to be $d + c_o t$, where t is the simulation time. We treat the fluid as inviscid and use a particle spacing of $\Delta x = 0.1$ unless stated otherwise in all our test cases. We use the results in the long domain as a reference and use this to compute the L_2 error in various properties at time t using

$$L_2(\phi) = \left(\frac{\sum_j (\phi(\mathbf{x}_j, t) - \phi_o(\mathbf{x}_j, t))^2}{\sum_n (\phi_o(\mathbf{x}_j, t))^2} \right)^{1/2}, \quad (5.20)$$

where ϕ is the property of interest at a particular timestep and ϕ_o is the corresponding property in the long domain.

2D backward-facing step

Backward-facing step is a classical problem to test the accuracy of open boundary condition implementations in SPH. For this problem, following the experimental work of Armaly et al. (1983), the step height is set as, $h = 4.9mm$ with inlet width $h_i = 5.2mm$. We compare the velocity profile at different stations with $x/h = 2.55, 3.57, 4.80, 7.14$ (where x is the distance downstream from the step). We compare our results with the

experimental results in Armaly et al. (1983). The Reynolds number of the flow is chosen to be 389 since the flow is no longer two-dimensional above this. In the simulation, we set $\rho = 1.225 \text{ kg/m}^3$ and $\nu = 1.47 \times 10^{-5} \text{ m}^2/\text{s}$ and calculate inlet velocity U using the relation $Re = 2Uh_i/\nu$. The schematic of the simulation model is shown in fig. 5.9.

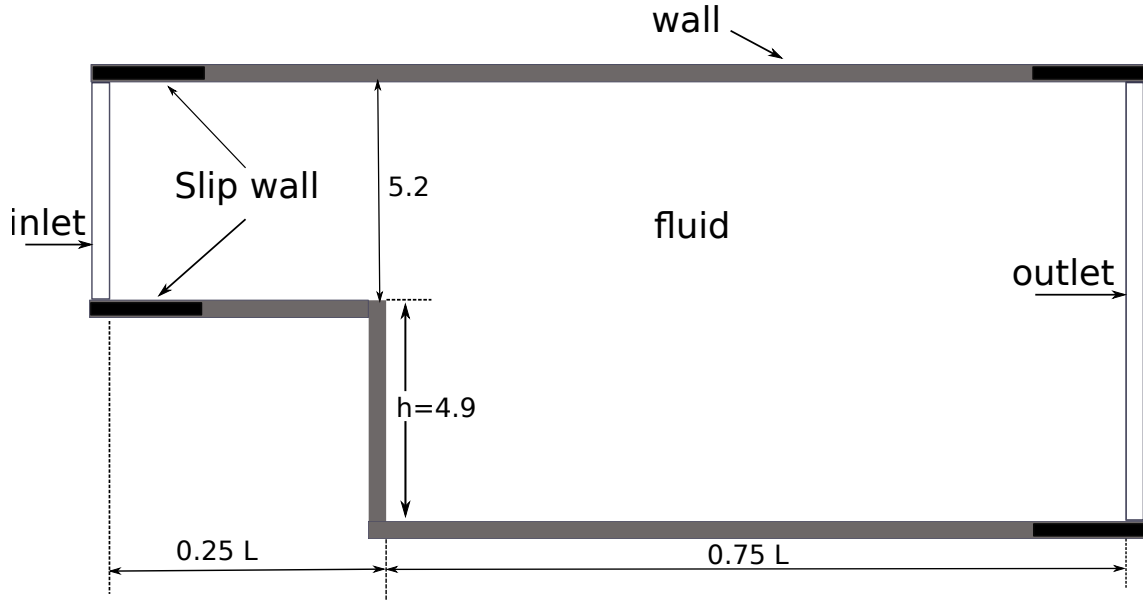


Figure 5.9 : Sketch of the domain used for backward-facing step simulations (all dimensions in mm).

At the walls, we satisfy the no-slip boundary condition. However, since the inlet is set at a constant velocity, a no-slip wall introduces non-physical pressure fluctuations. Thus a small part of the initial wall is set as a slip wall. Similarly, near the outlet, we allow slip at the wall in order to avoid vortices at the start of the flow. In this test, we have shown results for our proposed method and do-nothing only since the characteristic method and mirror methods failed to complete. In the case of the mirror method, the vortices reach the outlet and the simulation blows up. In the case of MOC, the criteria for reference parameter is not known. In fig. 5.10, we show the velocity profile for all the methods. It is evident from the plot that all the methods (hybrid, do-nothing, and modified do-nothing) are able to reproduce the results presented by Armaly et al. (1983).

Method	x_{rl}/h
Do-Nothing	8.030
Hybrid	8.009
New Do-Nothing	7.901

Table 5.1 : The reattachment length for $Re = 389$ for different outlet implementations.

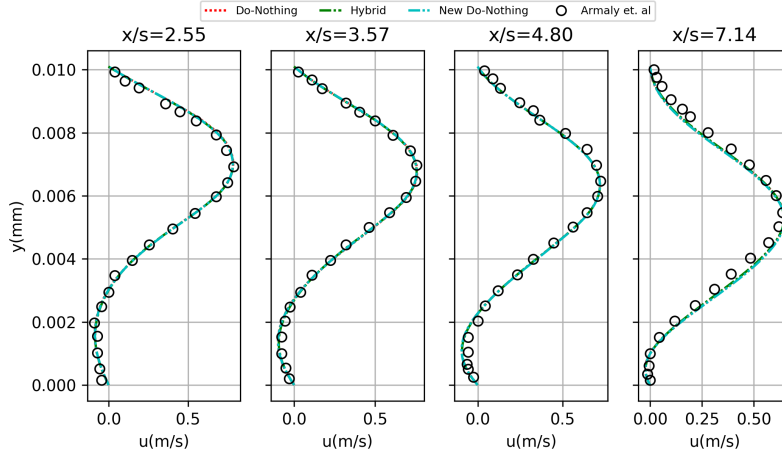


Figure 5.10 : Velocity at $t = 1.2 \text{ sec}$ for $Re = 389$ at different locations.

The reattachment length for the primary vortex is determined and presented in table 5.1. We can clearly see that the reattachment length is very close to the experimental value 7.94 from Armaly et al. (1983). This test case clearly shows that the proposed method shows very less difference from the experimental values compared to other methods. However, all the existing methods work equally well.

Flow past a 2D circular cylinder

The flow past a circular cylinder is a well-known benchmark to show the capability of inlet/outlet boundaries. We simulate this problem for all the methods described in this section. We consider a smaller domain compared to earlier research with fewer particles to show the effectiveness of the proposed method (Marrone et al., 2013; Tafuni et al., 2018). We consider a cylinder of diameter $D = 2m$. The channel width is $15D$ to avoid the effect of the wall and the length is $15D$, which is aligned along the x-axis. The cylinder is at $5D$ from the inlet interface as shown in fig. 5.11. Each inlet, outlet, and wall have 6 layers of particles which are enough to get full kernel support. The inlet is given a constant prescribed velocity, $U = 1m/s$. The walls act as a slip wall in order to avoid the effect of the boundary layer from the walls. The fluid properties, such as the kinematic viscosity of the flow, is evaluated using $\nu = UD/Re$, where Re is the Reynolds number of the flow and density $\rho = 1000kg/m^3$. We use a particle spacing $\Delta x = 0.0667$ and $h_{\Delta x} = 1.2$ which result in 201694 fluid particles in the domain. This spacing results in a cell Reynolds number of $Re_{cell} = Uh/\nu$ of 8. This suggests that this is a coarse simulation. In order to capture the curvature of the cylinder, we use the standard method as discussed in chapter 4.

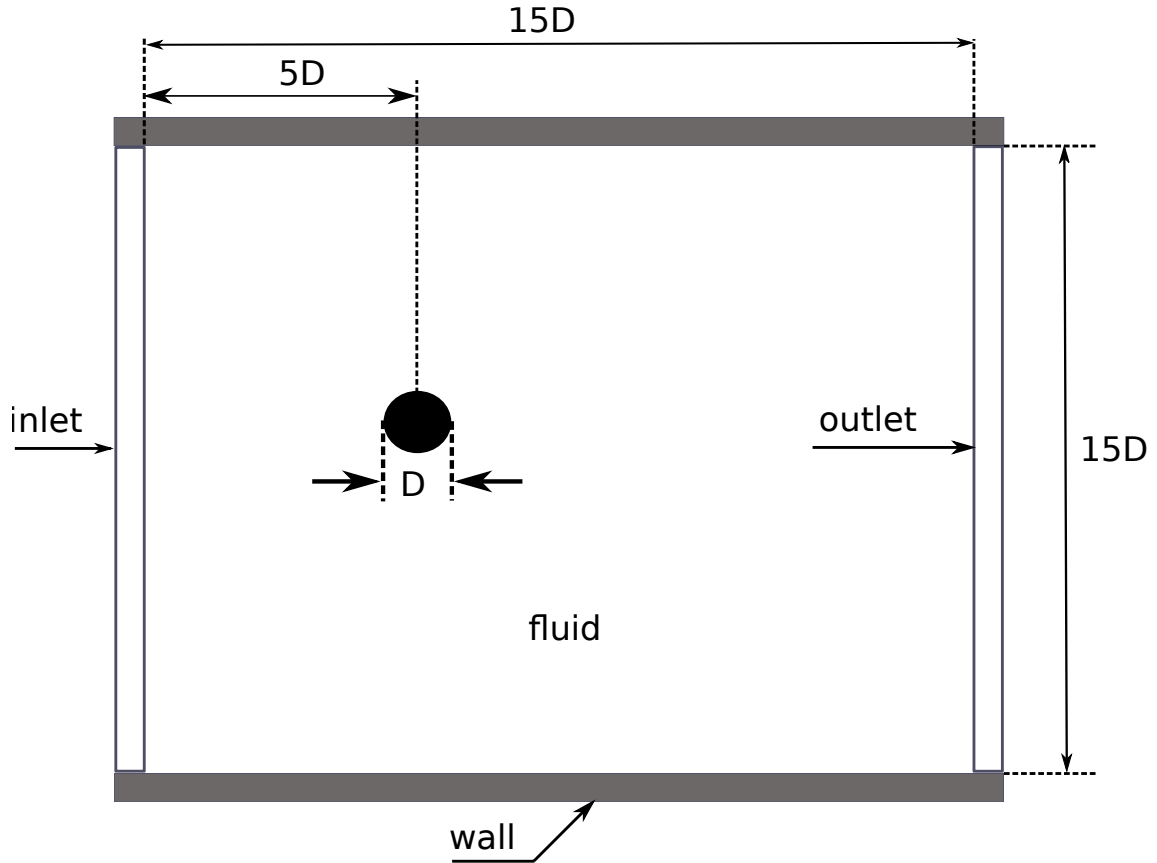


Figure 5.11 : Sketch of the domain used for the flow past a circular cylinder simulations.

We simulate the model for $Re = 200$ for all the methods. We normalize p and u measured at the probe such that $u^* = u/U$ and $p^* = \frac{2p}{\rho U^2}$ respectively. In fig. 5.12 and fig. 5.13, we show the pressure and velocity distribution at $t = 150s$, respectively, when the vortex shedding is well established. In the case of the mirror method, due to the high gradient near the outlet, spurious pressures arise, and the particle positions diverge. It is evident from the pressure plots in fig. 5.12 that the MOC reflects the pressure back into the domain when vortex shedding starts. In the case of do-nothing and modified do-nothing a significant increase in pressure of the domain is visible. Pressure for both hybrid and characteristic methods looks to be distributed around zero, which is essential for low numerical errors in pressure calculations. In fig. 5.13, all the methods show a similar pattern of the velocity field, and it is hard to comment on the relative merits of the methods.

In order to check the accuracy of the methods, we calculate the drag ($F_d = F_x$) and lift ($F_l = F_y$) forces on the cylinder for all the cases and evaluate the coefficient of drag, $c_d = F_d/(0.5\rho U^2)$, and lift $c_l = F_l/(0.5\rho U^2)$. A five-point average is taken to filter the noise. The force on the solid cylinder is determined by solving the momentum equation

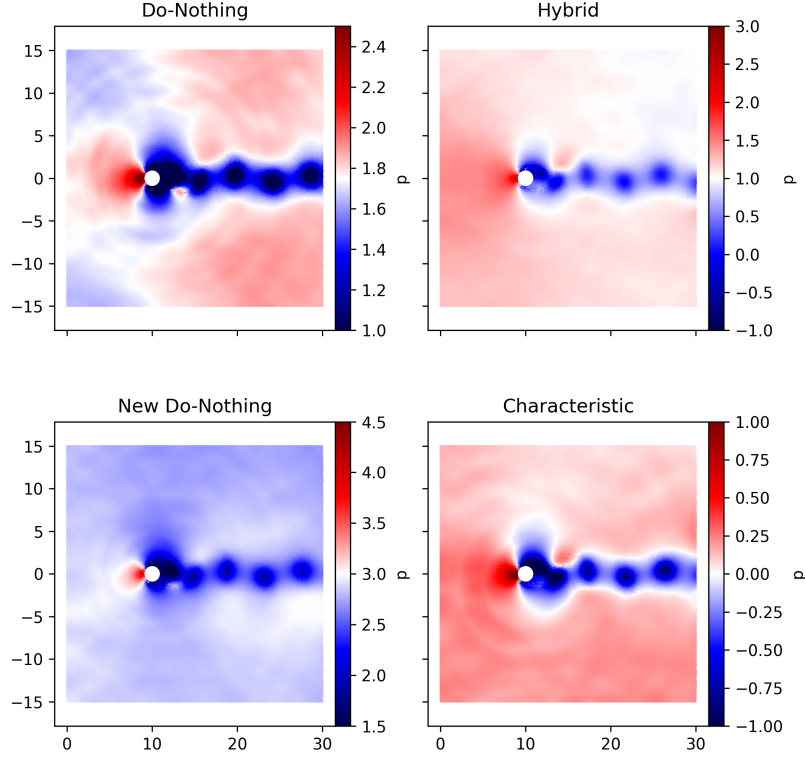


Figure 5.12 : Normalized pressure at $t = 150\text{sec}$ for $Re = 200$.

given by

$$\frac{F_{solid}}{m_{solid}} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}. \quad (5.21)$$

The above equation is discretized the same as done in the EDAC scheme. We also evaluate the Strouhal number $St = fD/U$ where f is the frequency of shedding. In fig. 5.14, we compare c_l for all the methods over time. The mirror method blows up after short time and generates a large back pressure. In the case of do-nothing and modified do-nothing, shedding starts earlier compared to hybrid and characteristic methods. In table 5.2, we compare c_d and c_l and St for all the methods and with results published by Guerrero (2009), Marrone et al. (2013), and Tafuni et al. (2018). We can see that in spite of having non-physical pressure variations in characteristic methods the value of c_d and c_l shows a close match. In the case of both do-nothing and modified do-nothing the values are close since the pressure increase of the domain is insignificant in the case of incompressible flows. In our proposed hybrid method, the pressure and velocity plots look similar to results published by Marrone et al. (2013) and Tafuni et al. (2018), the c_d and c_l are in the acceptable range presented in the literature.

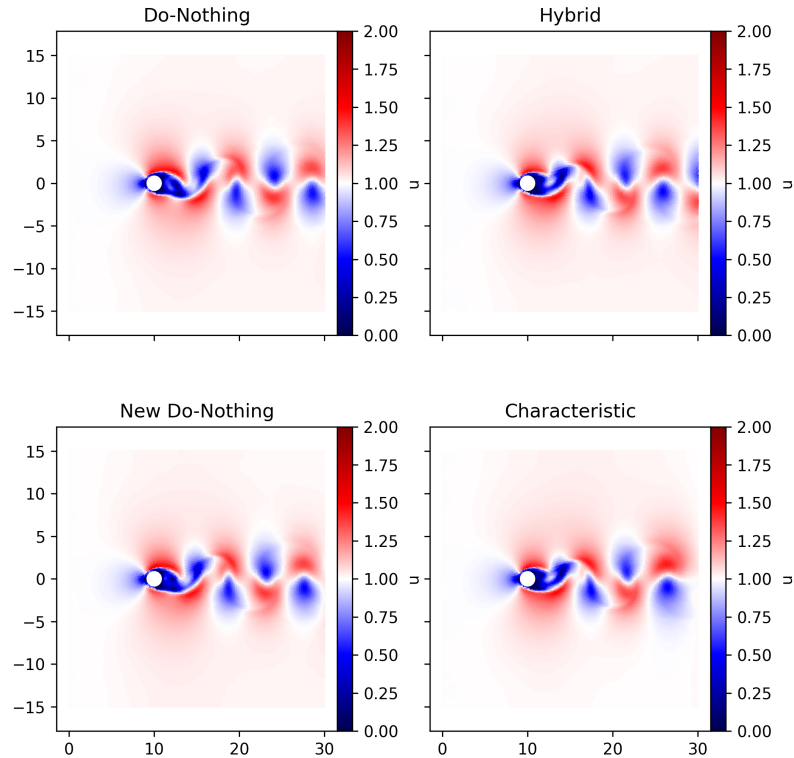


Figure 5.13 : Normalized velocity at $t = 150\text{sec}$ for $Re = 200$.

Method	c_d	c_l	St
Characteristic	1.494 ± 0.05	± 0.634	± 0.200
Do-Nothing	1.532 ± 0.05	± 0.744	± 0.210
Hybrid	1.524 ± 0.05	± 0.722	± 0.210
New Do-Nothing	1.540 ± 0.05	± 0.729	± 0.210
Marrone et al. (2013)	1.38 ± 0.05	± 0.680	0.200
Guerrero (2009)	1.409 ± 0.048	± 0.725	-
Tafuni et al. (2018)	1.46	± 0.693	0.206

Table 5.2 : Comparison of c_l , c_d and St values for different methods for $Re = 200$.

In the previous two test cases, we demonstrated that all the boundary condition implementations show accurate results irrespective of the absence of the non-reflective nature. Therefore, these test cases do not test the essential non-reflective nature of an outlet boundary implementation. In the next section, we use existing and propose new test cases that test various aspects of an outlet boundary implementation.

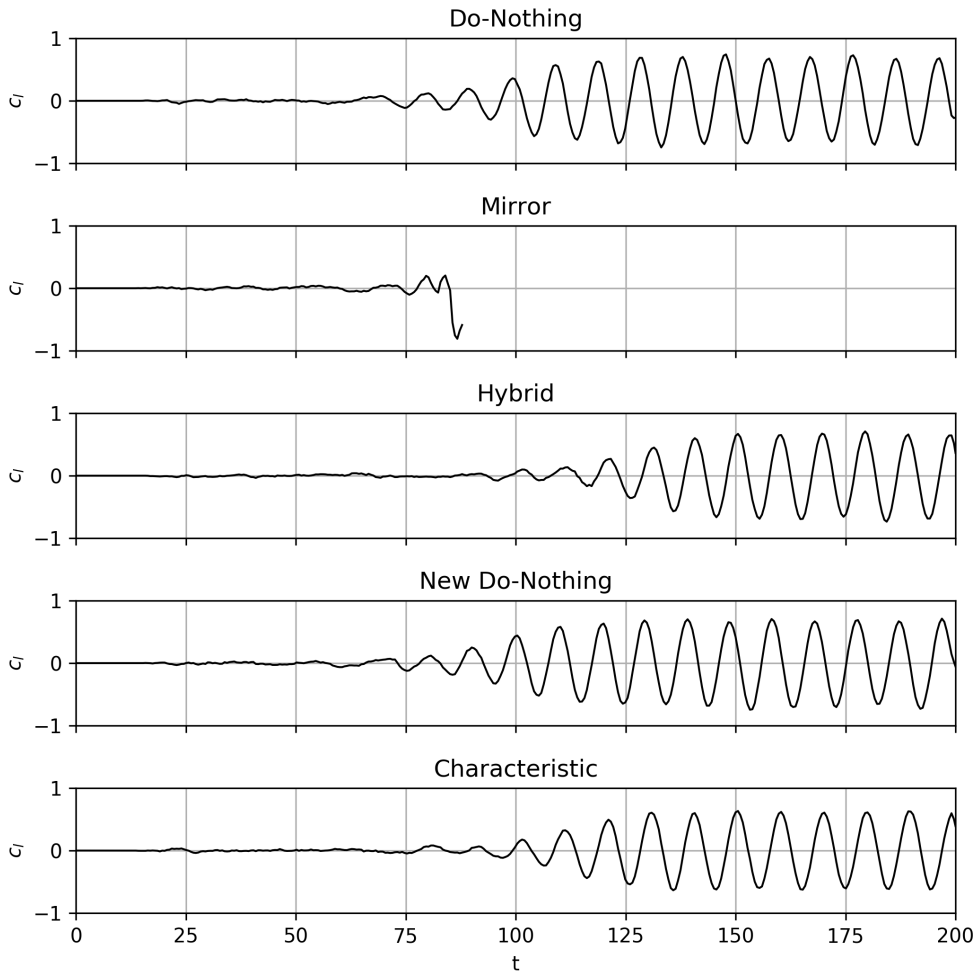


Figure 5.14 : Plot for c_d for all methods at $Re=200$

1D Pressure bump

This test case was proposed by Lastiwka et al. (2009). In this test case, the fluid domain is initialized with a pressure variation given by

$$p(x) = 1.0 - 0.2e^{-\frac{(x-0.5)^2}{0.001}}. \quad (5.22)$$

The pressure at the inlet and outlet is initialized with $p = 1.0$. The velocity of the domain, including inlet and outlet, remains constant ($=1m/s$) for all times. The domain length is $1m$, and the pressure bump is at $x = 0.5m$. We use the artificial viscosity parameter, $\alpha = 0.1$, as mentioned in Lastiwka et al. (2009). We simulated the test case for all the outlet boundary implementations described in the previous section. In the case of the MOC for all the test cases u_o, p_o and ρ_o is taken as $1.0m/s, 1.0 Pa$ and $1000kg/m^3$

respectively. In fig. 5.15, we compare the pressure along the centerline of the domain at different times for all the methods. It can be seen that the mirroring technique results in a significant drop in pressure towards the end. The modified do-nothing increases the pressure in the domain by a small amount. All other cases match well with the MOC and the long domain.

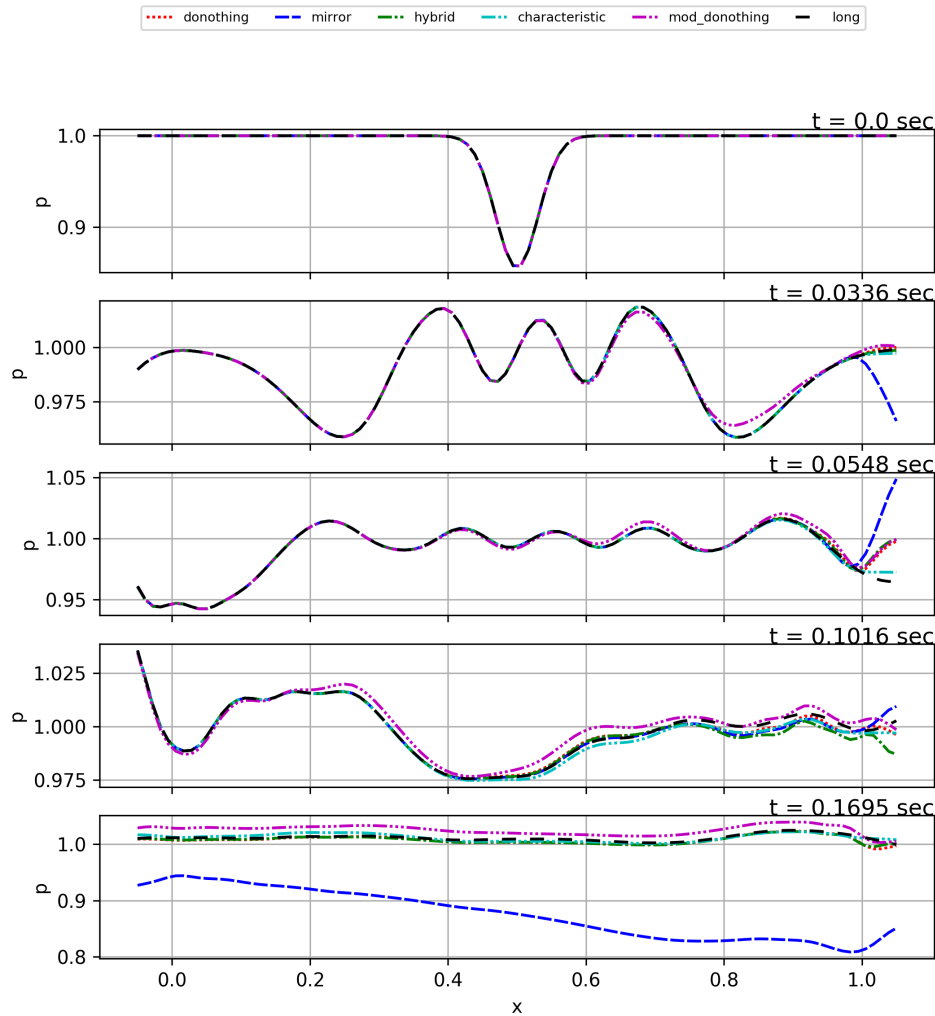


Figure 5.15 : Pressure plot at various times for the different methods. The solid line denotes the solution with the long domain.

2D pulse

This benchmark tests the non-reflectivity for a two-dimensional disturbance. A 2D domain is considered, consisting of fluid with domain length $L = 2m$ and width $W = 2m$. The probe is placed at $d = 1.7m$ from the inlet. The fluid region is constrained by inviscid walls on both sides. The inflow is taken from the left, and the outlet is kept at the right of

the fluid. The inlet, wall, and outlet are initialized with 6 layers of particles. In order to introduce a 2D variation, the x -component of the velocity is made a function of y , given by

$$u(x, y, t) = \begin{cases} 1.0 + 0.5 \cos\left(\frac{\pi y}{12}\right) e^{\frac{(t-1)^2}{\delta}} & 1.0 < t < 1.1, \\ 1.0 & \text{elsewhere.} \end{cases} \quad (5.23)$$

Figure 5.16 shows the plot of u^* and p^* versus time for the different outlets and table 5.3 shows L_2 errors in the pressure and velocity for the different outlet implementations.

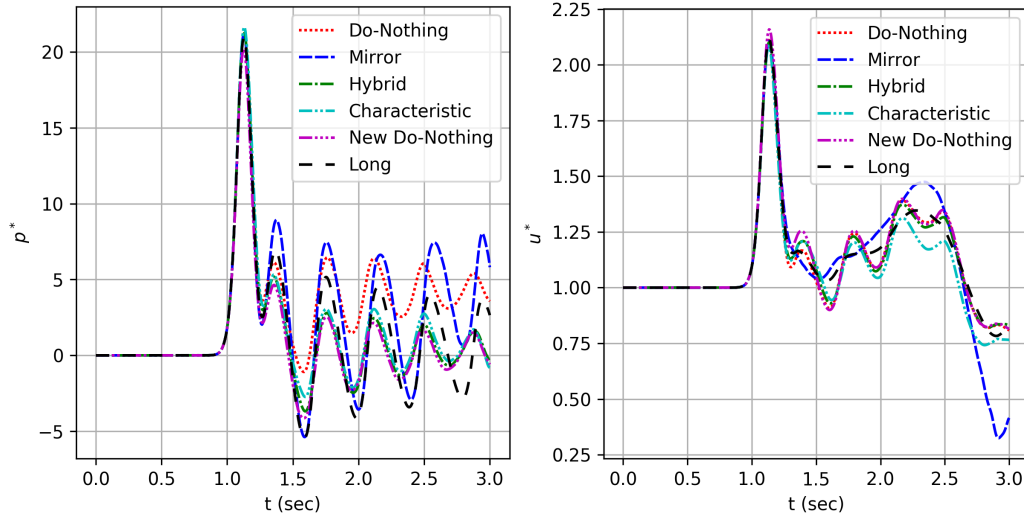


Figure 5.16 : Normalized pressure (left) and velocity (right) plots at $x = 1.7m$ with time for 2D varying inlet.

The pressure variation with the MOC and hybrid methods are very close to the results for a long domain compared to the mirror and do-nothing methods. It can be seen that the mirroring technique generates a lot of reflections into the fluid as compared to do-nothing and MOC. In the case of the do-nothing, a significant increase in pressure can be seen just after the wave passes through the outlet (at around 1.25s).

Methods	$L_2(p^*)$	$L_2(u^*)$
Characteristic	0.328	0.057
Do-Nothing	0.629	0.038
Hybrid	0.311	0.035
Mirror	0.409	0.106
New Do-Nothing	0.341	0.042

Table 5.3 : L_2 error in the p^* and u^* measured at the probe for the 2D pulse problem.

Looking at the variation of the velocity, we can see that both modified do-nothing and new hybrid method show a close match to the results for a long domain. After 2s, the

MOC method differs from the long-domain results due to the spatial variations arriving near the outlet. The modified do-nothing method is clearly better than the standard do-nothing scheme. These conclusions are also borne out by the values of the L_2 norm as seen in table 5.3. The proposed hybrid method has the least errors.

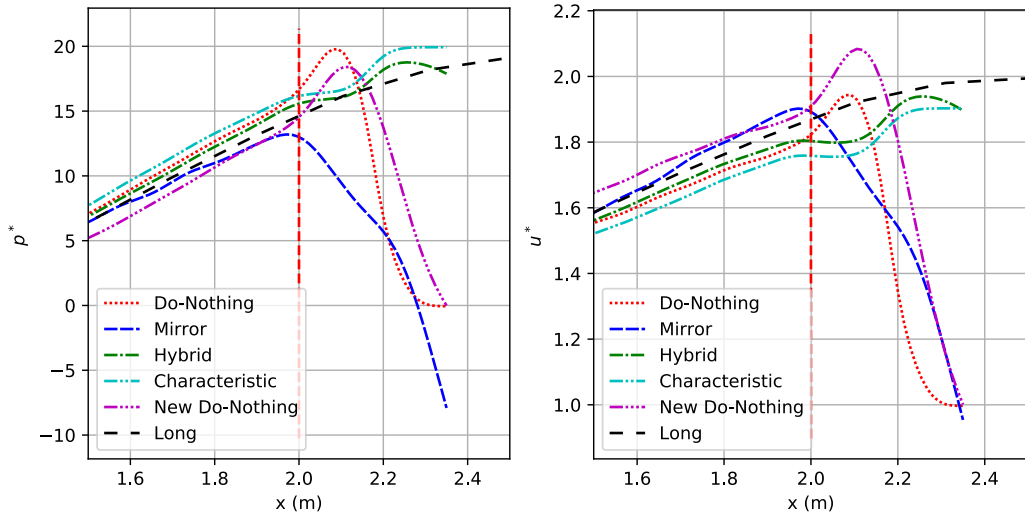


Figure 5.17 : Normalized pressure (left) and velocity (right) along the $y = 0$ line for the 2D pulse problem. The left of the dashed red line is fluid, and the right is outlet region.

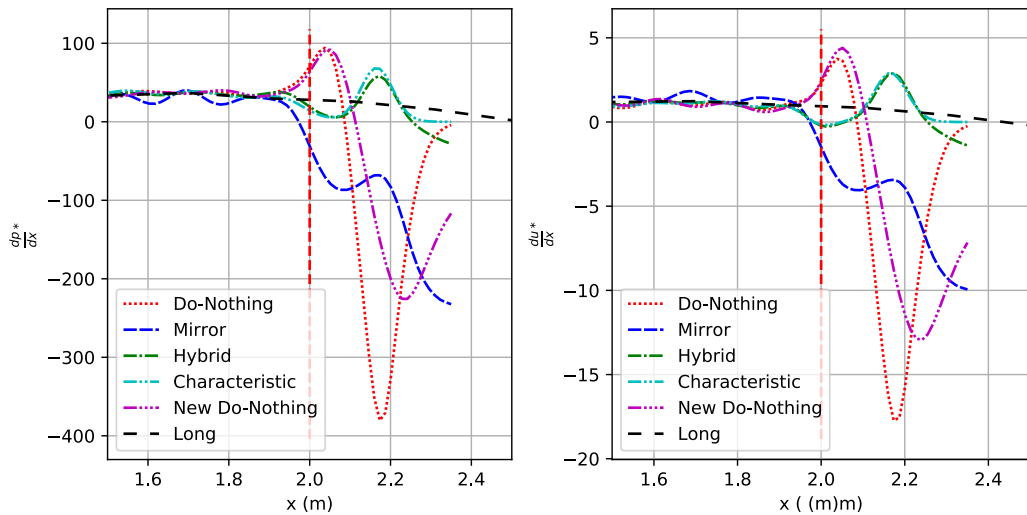


Figure 5.18 : Normalized pressure (left) and velocity gradients (right) along the $y = 0$ line for the 2D pulse problem. Left of the dashed red line is fluid and right is outlet region.

In order to show the nature of the property variation across the fluid outlet interface due to extrapolation, we interpolate pressure, velocity and their gradients on a $y = 0$ line as shown in fig. 5.17 and fig. 5.18. In the case of the mirror method, the gradient of the

property is zero at the interface. The property is mirrored about the domain boundary; however, both hybrid and do-nothing retain the history of the particle such that velocity and pressure in the outlet does not affect the upstream flow. On looking at the gradient along x of the property for all the methods in fig. 5.18, we find that the mirroring technique impose natural boundary conditions on fluid particles near the outlet i.e $\partial u/\partial x = 0$, and $\partial p/\partial x = 0$. In case of do-nothing and modified do-nothing, the velocity and pressure profiles matched the long domain but the gradient changes significantly. However, the method of characteristics and hybrid maintains the flow gradients along with the flow variables as they are. In the context of the SPH, the latter seems to be very important.

As discussed in appendix A.5.4, the EDAC method involves a parameter called α , which increases the pressure damping. We explore varying the parameter α and study the error in p^* for the different schemes in table 5.4. It can be observed that as α increases, the pressure oscillations are reduced, and therefore, the errors reduce for all the schemes. However, the greatest reduction is for the original do-nothing and mirror methods. The others are not significantly affected. This suggests that the hybrid method and modified do-nothing are robust techniques.

Methods	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 1.0$
Characteristic	0.336	0.334	0.328	0.315
Do-Nothing	1.135	0.864	0.629	0.587
Hybrid	0.339	0.319	0.311	0.306
Mirror	0.533	0.463	0.409	0.371
New Do-Nothing	0.391	0.543	0.341	0.354

Table 5.4 : L_2 error in p^* measured at the probe for the 2D pulse with the change in EDAC parameter α .

1D ramp

In this test case, we impose a ramp velocity on the inlet particles such that $u = 0m/s$ at $t = 0s$ and $u = 1m/s$ at $t = 1s$. After time $t = 1s$, we fix the velocity at $1m/s$. The size of the domain, boundary conditions, and initialization are the same as in the case of the 2D pulse. We simulate the test case for each method and compare it with results for a long domain. In fig. 5.19, we plot the p^* and u^* in the domain and tabulate the L_2 errors in table 5.5. In the case of pressure, the hybrid, mirror, and modified do-nothing methods work well. The standard do-nothing method generates a significantly high pressure as the initial particles at the outlet do not move and thereby cause an increase in pressure. In the

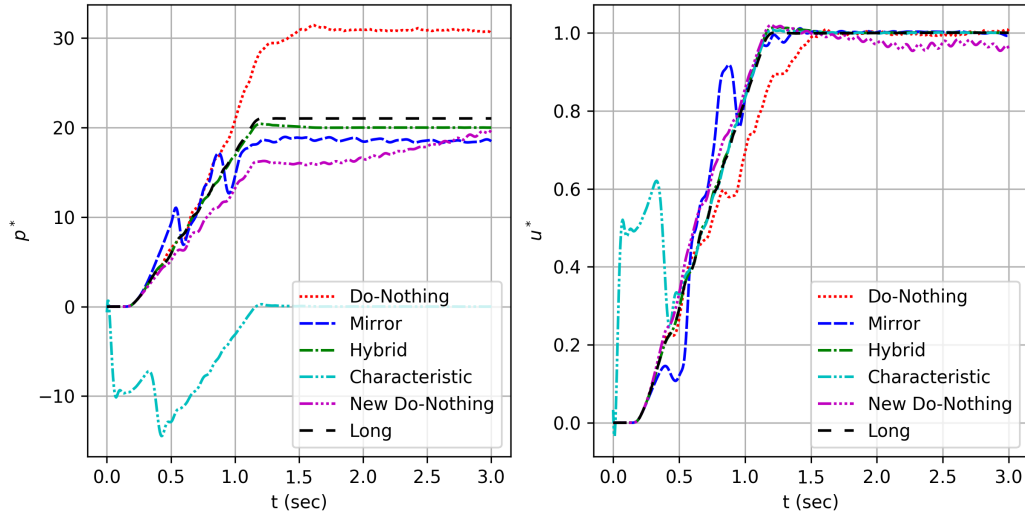


Figure 5.19 : Normalized pressure (left) and velocity (right) plots at $x = 1.7m$ with time for ramp inlet.

case of the MOC, there is no specific method to determine the reference values u_o and p_o at the initial stage, and this seems to cause the problems. Similar issues are seen in the case of the velocity for the MOC. As seen in table 5.5, the hybrid method has the least errors for both pressure and velocity.

Methods	$L_2(p^*)$	$L_2(u^*)$
Characteristic	1.099	0.193
Do-Nothing	0.433	0.066
Hybrid	0.043	0.007
Mirror	0.123	0.074
New Do-Nothing	0.197	0.039

Table 5.5 : L_2 error in the p^* and u^* measured at the probe for the ramp velocity problem.

2D vortex

In this test case, a vortex is generated in the inlet moving with a constant velocity of $1m/s$ and allowed to go through the outlet. This case tests the outlet for permeability for a velocity variation similar to vortex shedding. It is important that this be preserved for most engineering flow simulations. The domain size is kept the same as in the case of the 2D pulse; however the width is doubled to accommodate the vortex. The vortex is generated by changing the velocity at the inlet with time using

$$(u, v) = \left(1.0 + \frac{\Gamma y}{r^2 + 0.2}, \frac{-\Gamma x}{r^2 + 0.2} \right), \quad (5.24)$$

where $\Gamma = 0.1$ is the vortex strength, and $r = \sqrt{x^2 + y^2}$ is the distance from the center of the vortex. In order to calculate the distance of the vortex, we set

$$x = u(1 - t), \quad (5.25)$$

where u is the speed of the vortex and 1.0 is the initial distance of the vortex center from the beginning of the inlet. We test the vortex advection with the methods and compare them with the results for a long domain.

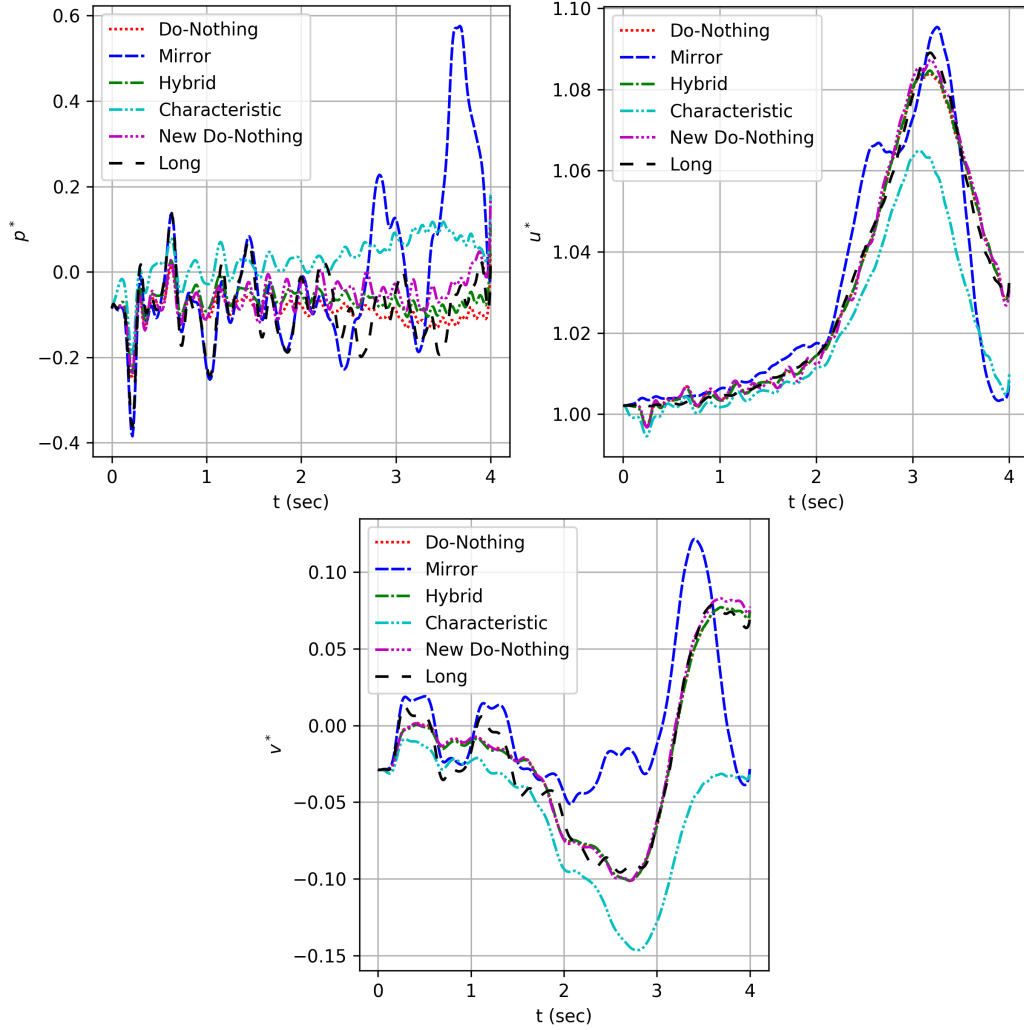


Figure 5.20 : Normalized pressure (left), u-velocity (right) and v-velocity (center) plots at $x = 1.7m$ with time for 2D vortex advection with $1m/s$.

In fig. 5.20, we have plot the p^* , u^* , and v^* for the different methods. In table 5.6, we tabulate the L_2 errors of the pressure and velocity. It is evident from the plots that the do-nothing, modified do-nothing, and our new hybrid method match the results of a long domain well. However, in the case of mirror method, a lot of back pressure fluctuation is visible. The MOC shows a significant deviation from the long domain and also does not preserve the velocity variation. In the pressure plot, we can see that the mirror method

Methods	$L_2(p^*)$	$L_2(u^*)$	$L_2(v^*)$
Characteristic	1.216	0.013	0.959
Do-Nothing	0.554	0.002	0.174
Hybrid	0.569	0.002	0.173
Mirror	1.588	0.010	0.806
New Do-Nothing	0.629	0.002	0.182

Table 5.6 : L_2 error in the p^* , u^* , and v^* measured at the probe for the moving vortex problem, with no viscosity.

Methods	$L_2(p^*)$	$L_2(u^*)$	$L_2(v^*)$
Characteristic	0.556	0.001	0.263
Do-Nothing	0.521	0.001	0.258
Hybrid	0.519	0.001	0.255
Mirror	0.475	0.006	0.725
New Do-Nothing	0.556	0.001	0.263

Table 5.7 : L_2 error in the p^* , u^* , and v^* measured at the probe for the moving vortex problem, with $Re = 100$.

Methods	$L_2(p^*)$	$L_2(u^*)$	$L_2(v^*)$
Characteristic	1.213	0.013	0.961
Do-Nothing	0.553	0.002	0.175
Hybrid	0.566	0.002	0.173
Mirror	1.541	0.010	0.803
New Do-Nothing	0.589	0.002	0.181

Table 5.8 : L_2 error in the p^* , u^* , and v^* measured at the probe for the moving vortex problem, with $Re = 10000$.

shows a perfect match before the vortex reaches the probe. Thus it is suitable for outlets with very low gradients. However, once the vortex reaches the probe, the results of the mirror method are very poor. The L_2 errors clearly show that the do-nothing methods and the hybrid schemes work well. In particular, the error in p and v is high for the mirror and method of characteristic.

As mentioned earlier, all our simulations are inviscid, which suggests an infinite Reynolds number. However, to investigate the effect of the Reynolds number on the outlet, we performed the above simulation at $Re = 100$ and 10000 . In table 5.7 and table 5.8, we show the errors for $Re = 100$ and 10000 , respectively. The hybrid method and do-nothing have low errors compared to other methods. Furthermore, as the Reynolds number reduces, the fluid becomes increasingly viscous; therefore, the errors in the method of characteristics and the mirror method reduce. This shows the importance of the new method.

5.3 Summary

In this chapter, we review the established techniques for implementing various boundary conditions in the context of weakly-compressible SPH schemes. In the case of open boundary conditions, non-reflectivity is very important. We show that the traditional test cases, like flow past a circular cylinder and backward-facing step, are unable to capture this property. In order to systematically examine the NRBC, we construct four simple test problems. These tests clearly show the deficiencies of the existing approaches.

- Do-nothing method is only suitable for problems where high-intensity acoustic pressure waves are absent.
- The mirror method works best for flows where the gradients are very low near the outlet.
- The MOC shows excellent results where reference properties are known a priori but are not very effective when there are gradients in the flow at the exit.

Based on this, we propose a new generalized scheme which combines the do-nothing and characteristic-based outlet into a new hybrid technique. The proposed technique works well with both high-intensity acoustic waves and high-gradient flow near the outlet. Unlike the MOC, it calculates reference flow variables by time averaging. We also propose a simpler and slightly modified do-nothing boundary condition that produces good results.

The test cases considered in this chapter simulate an actual fluid flow. However, in the MMS, we can initialize the field using any suitable field to limit the test to a particular aspect of the implementation. In the next chapter, we discuss the procedure one should follow to manufacture a solution to verify the boundary condition implementations discussed in this chapter. We use these manufactured solutions to verify the convergence of these boundary condition implementations.

Chapter 6

Verification of boundary condition implementations

In this chapter, we construct various manufactured solutions (MSs) to satisfy the boundary condition (BC) on different domain shapes. We propose different MSs to verify Neumann pressure, slip, and no-slip boundary conditions for straight, convex, and concave boundary surfaces. We also propose MSs for inflow and outflow boundary conditions with and without a wave traveling through the interface. Bond et al. (2007) and Choudhary (2015) proposed a method to construct MS for boundary condition verification of mesh-based codes. Assuming an MS for a periodic domain given by $\phi(x, y) = \phi_o + \tilde{\phi}(x, y, t)$. In order to obtain an MS for a boundary surface given as $C(x, y) = b$, we multiply the original MS with $(b - C(x, y))^m$. We write the new MS as

$$\phi_{BC}(x, y, t) = \phi_o + (b - C(x, y))^m \tilde{\phi}(x, y, t), \quad (6.1)$$

where m is the order of the boundary condition. For example, for the Dirichlet boundary $m = 1$ and for Neumann boundary $m = 2$.

We use these MSs to verify the convergence of various widely used boundary condition implementations. We identify the boundary implementations that show convergence for all the boundary shapes. We finally propose a complete second-order convergent weakly compressible scheme that implements all the boundary conditions to simulate a flow past an arbitrarily shaped obstacle. We solve the flow past a circular cylinder in order to demonstrate the accuracy compared to ISPH and EDAC schemes. In the next section, we construct MS for different boundary conditions using the method described above, modified for SPH schemes.

6.1 Manufactured solutions for BCs

The solid boundary can be straight or curved. In this work, we do not consider non-smooth geometric features, like a corner. For corners, one requires a discontinuous MS, and at a discontinuity, the higher-order terms fail to show the actual order of convergence. However, the method showing second-order convergence for smooth boundary will perform better than other methods. We consider three types of boundary shapes viz. straight, convex, and concave as shown in fig. 6.1. The fluid particles are represented by the blue color, and the green color particles represent the ghost particles for which we set the properties using the MS. The particles colored in orange are used to verify a particular method. In the domain referred to as ‘straight’, the ghost particles in orange have a constant normal. The domain referred to as ‘convex’, the boundary is a convex surface, whereas the domain referred to as ‘concave’, the boundary is a concave surface. We note that both convex and concave domains are identical; however, the boundary surfaces of interest are different.

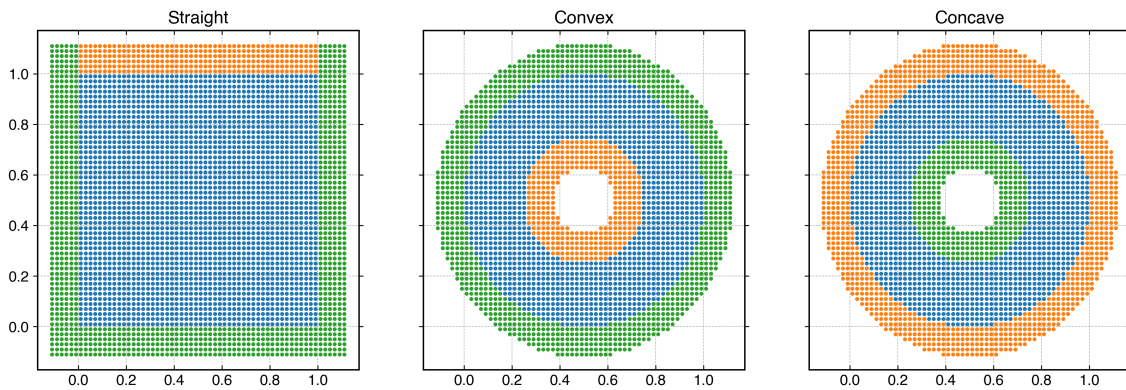


Figure 6.1 : Types of domains considered to test the convergence of solid boundary implementation. The fluid particles are represented by the blue color, the particles in green represent ghost particles on which properties are set using MS, and the particles in orange are used to test the convergence of boundary implementation of interest.

We observe that the convex and concave domains have a staircase pattern at the boundary due to the use of the Cartesian arrangement of particles to represent the boundary. We use the Hybrid packing discussed in section 4.1.3, to pack the particles near both inner and outer surfaces. In fig. 6.2, we plot the packed version of the convex and concave domains, and we refer to these as ‘packed-convex’ and ‘packed-concave’, respectively.

In order to test the open-boundary condition implementation, we require the inlet and outlet boundary to continuously add and remove particles from the domain, respectively. Furthermore, the inlet and outlet condition requires that the flow is only along the

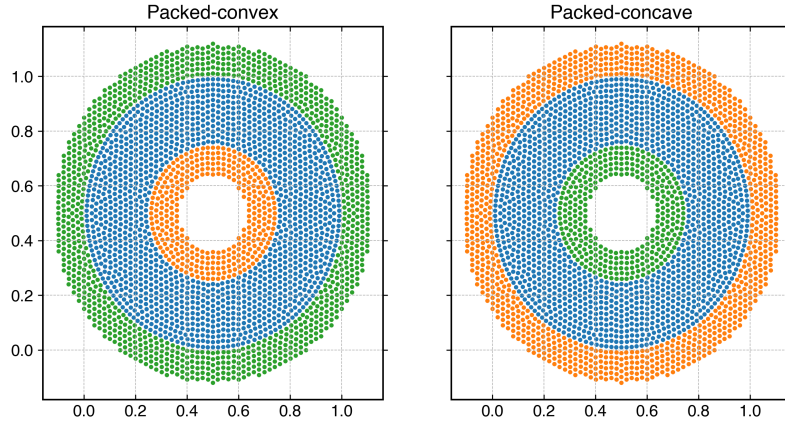


Figure 6.2 : The packed version of convex and concave domains. The fluid particles are represented by blue color, the particles in green represents ghost particles on which properties are set using MS, and the particles in orange are used to test the convergence of desired boundary condition implementation.

normal at the boundary. In order to satisfy these conditions, we use a $1m \times 1m$ domain, with an inlet and outlet on the left and right, respectively. In fig. 6.3, we show the domain with ghost particles representing the inflow (in green), outflow (in red), and wall (in orange).

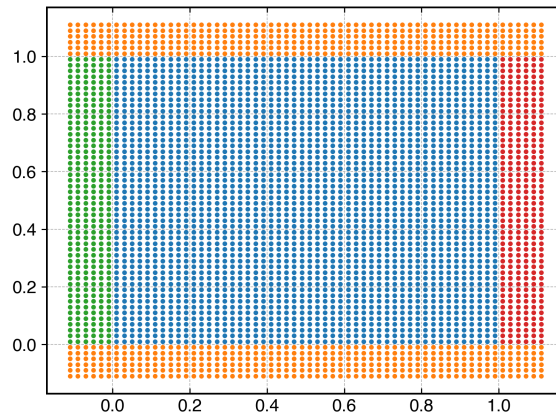


Figure 6.3 : The domain used for the verification of inlet and outlet boundary implementation. The blue particles represent the fluid, orange particle represent the wall, green particles are inflow particles, and red particles are the outflow particles.

We take the following steps in order to construct an MS for a boundary surface $C(x, y) = b$:

1. Construct a base MS such that it satisfies the general requirement of MMS (see section 3.2).

2. In the context of SPH, we additionally require the velocity field to be non-solenoidal as discussed in section 3.3.
3. Modify the property of interest such that the boundary condition at $C(x, y) = b$ is satisfied. For example, for the no-slip boundary, only velocity needs to be modified.
4. We note that one should ensure that the MS of the property of interest is non-zero on the boundary before the boundary condition is satisfied. For example, if we need $\mathbf{u} \cdot \hat{\mathbf{n}}$ at the boundary to be zero, we must make sure that $\mathbf{u} \neq 0$ at the boundary.

In the next sections, we use the above procedure to construct MS for Neumann pressure, slip, no-slip, and inflow and outflow boundary conditions.

6.1.1 Neumann pressure boundary

In this boundary condition, we ensure that $\nabla p \cdot \hat{\mathbf{n}} = 0$, where $\hat{\mathbf{n}}$ is normal to the boundary surface. For the straight domain the normal $\hat{\mathbf{n}} = \mathbf{j}$. Therefore, we can construct the MS given by

$$\begin{aligned} u(x, y) &= (y - 1) \sin(2\pi x) \cos(2\pi y), \\ v(x, y) &= -(y - 1) \sin(2\pi y) \cos(2\pi x), \\ p(x, y) &= x^2 + \cos(4\pi x). \end{aligned} \tag{6.2}$$

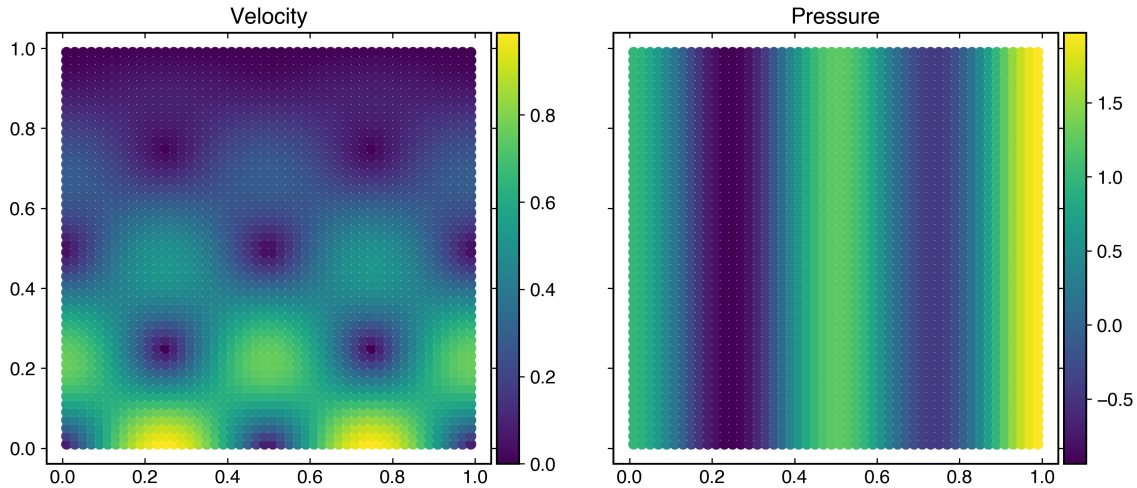


Figure 6.4 : Velocity and pressure contours on the straight domain in fig. 6.1 of the MS in eq. (6.2).

In fig. 6.4, we show the contour plot of the above MS in the straight domain. In the case of the convex domain, the normal to the surface is given by $\hat{\mathbf{n}} = (x - 0.5)\mathbf{i} + (y - 0.5)\mathbf{j}$. Therefore, we can construct an MS given by

$$\begin{aligned}
u(x, y) &= (y - 1) \sin(2\pi x) \cos(2\pi y), \\
v(x, y) &= -(y - 1) \sin(2\pi y) \cos(2\pi x), \\
p(x, y) &= \tan^{-1} \left(\frac{(y - 0.5)^2}{(x - 0.5)^2} \right).
\end{aligned} \tag{6.3}$$

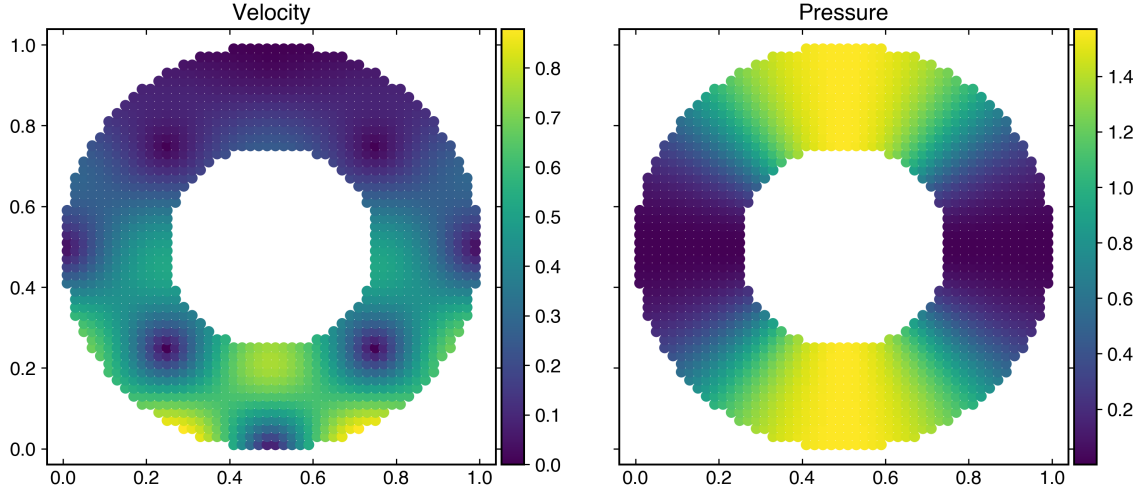


Figure 6.5 : Velocity and pressure contours on the convex/concave domain of the MS in eq. (6.3).

In fig. 6.5, we show the contour plot of the above MS in the convex domain. Since, for the concave domain, the normal remains the same, we can use the same MS since it satisfies $\nabla p \cdot \hat{\mathbf{n}} = 0$ at the surface of interest. We note that for the packed version of the domain in fig. 6.2, we can use the same MS as used in the unpacked version as the surface of interest is exactly the same.

6.1.2 Slip boundary condition

For the slip boundary condition, we ensure that $\mathbf{u} \cdot \mathbf{n} = 0$ at the boundary surface. For the straight domain, we construct the MS given by

$$\begin{aligned}
u(x, y) &= (y - 1) \sin(2\pi x) \cos(2\pi y) + 1, \\
v(x, y) &= (y - 1)^2 \sin(2\pi y), \\
p(x, y) &= \cos(4\pi x) + \cos(4\pi y).
\end{aligned} \tag{6.4}$$

In fig. 6.6, we plot the velocity and pressure contour generated by the MS in eq. (6.4). For the convex domain the normal $\hat{\mathbf{n}} = (x - 0.5)\mathbf{i} + (y - 0.5)\mathbf{j}$, therefore we construct the

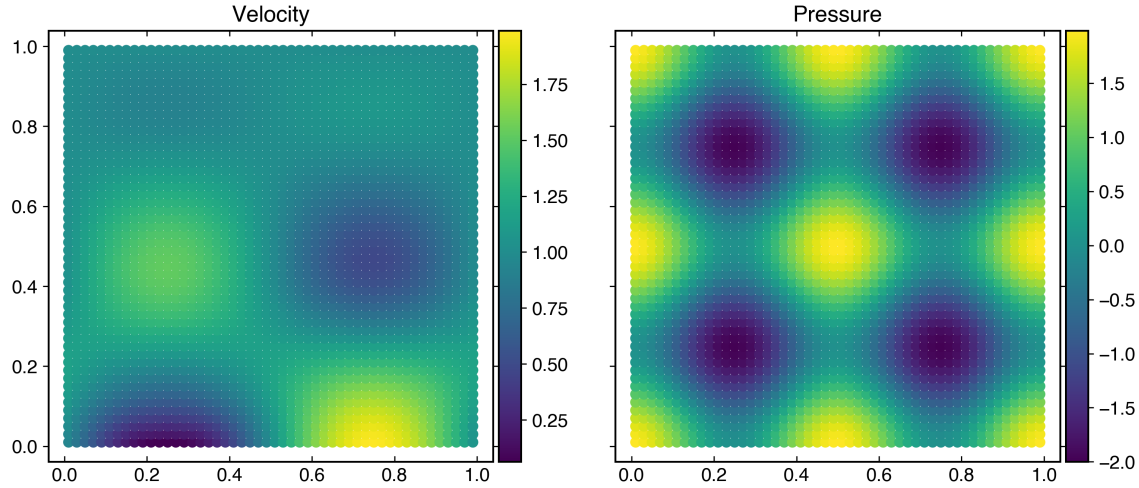


Figure 6.6 : Velocity and pressure contours on the straight domain of the MS in eq. (6.4).

MS given by

$$\begin{aligned}
 u(x, y) &= (y - 0.5) \sin(2\pi x) \cos(2\pi y), \\
 v(x, y) &= -(x - 0.5) \sin(2\pi x) \cos(2\pi y), \\
 p(x, y) &= \cos(4\pi x) + \cos(4\pi y),
 \end{aligned} \tag{6.5}$$

such that $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$. In fig. 6.7, we plot the velocity and pressure contour generated from the MS in eq. (6.4). We note that since for the concave as well as packed domains, the normal remains the same; therefore, we can use the same MS in eq. (6.5) for all these domains.

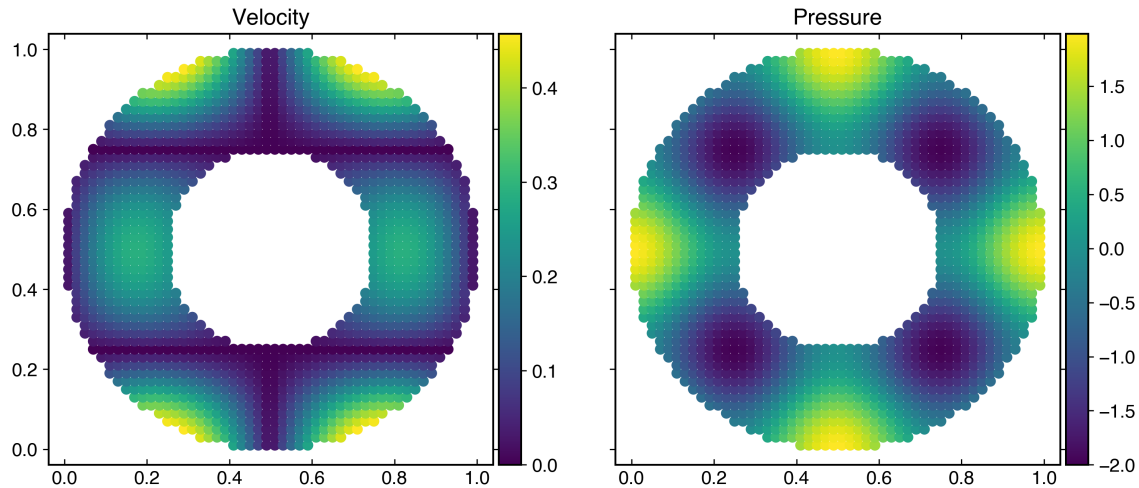


Figure 6.7 : Velocity and pressure contours on the convex/concave domain of the MS in eq. (6.5).

6.1.3 No-slip boundary condition

For a no-slip boundary condition, we ensure that the velocity at the surface is zero for a stationary wall. For the straight domain, we construct the MS given by

$$\begin{aligned} u(x, y, t) &= (1 - y)^2 e^{-10t} \sin(2\pi x) \cos(2\pi y), \\ v(x, y, t) &= -(1 - y)^2 e^{-10t} \sin(2\pi y) \cos(2\pi x), \\ p(x, y, t) &= (\cos(4\pi x) + \cos(4\pi y)) e^{-10t}. \end{aligned} \quad (6.6)$$

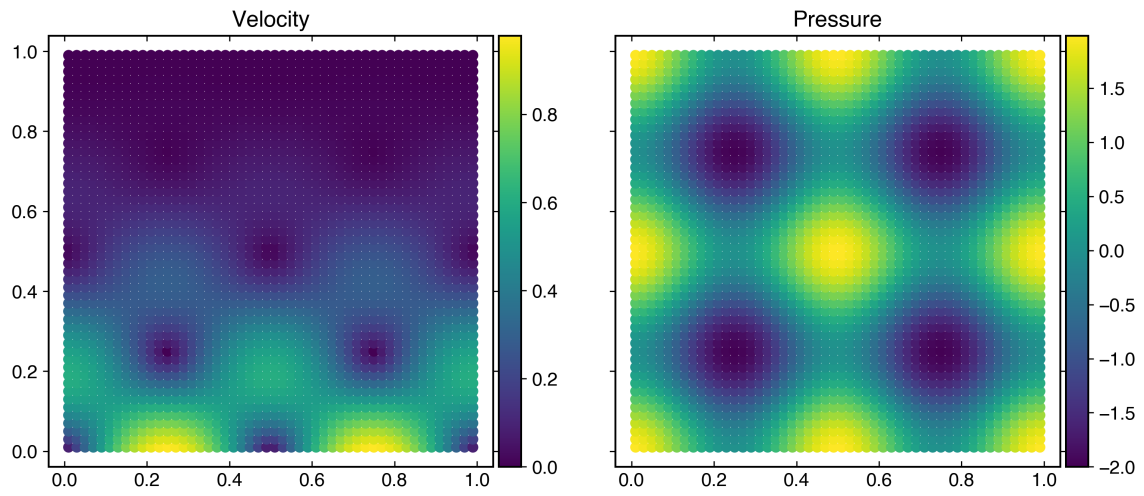


Figure 6.8 : Velocity and pressure contours on the straight domain of the MS in eq. (6.6).

In fig. 6.8, we plot the contour plot for the velocity and pressure generated by the MS in eq. (6.6). In order to construct an MS for the convex domain in fig. 6.1, we construct the MS such that the velocity is zero on the inner surface of the domain, given by

$$\begin{aligned} u(x, y, t) &= \left(-(x - 0.5)^2 - (y - 0.5)^2 + 0.0625 \right) \\ &\quad e^{-10t} \sin\left(\pi\left(2(x - 0.5)^2 + 2(y - 0.5)^2\right)\right), \\ v(x, y, t) &= -\left(-(x - 0.5)^2 - (y - 0.5)^2 + 0.0625 \right) \\ &\quad e^{-10t} \cos\left(\pi\left(2(x - 0.5)^2 + 2(y - 0.5)^2\right)\right), \\ p(x, y, t) &= (\cos(4\pi x) + \cos(4\pi y)) e^{-10t}. \end{aligned} \quad (6.7)$$

In fig. 6.9, we plot the velocity and pressure contour generated from the MS in eq. (6.7). In order to construct the MS for the concave domain in fig. 6.1, we make the velocity zero on the outer surfaces given by

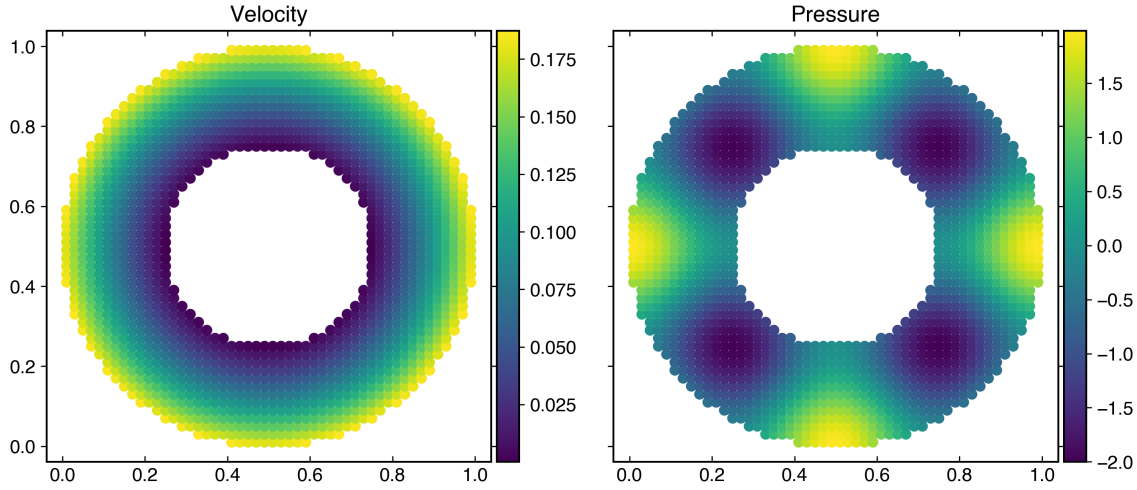


Figure 6.9 : Velocity and pressure contours on the convex domain of the MS in eq. (6.7).

$$\begin{aligned}
 u(x, y, t) &= \left(-(x - 0.5)^2 - (y - 0.5)^2 + 0.25 \right) \\
 &\quad e^{-10t} \sin \left(\pi \left(2(x - 0.5)^2 + 2(y - 0.5)^2 \right) \right), \\
 v(x, y, t) &= - \left(-(x - 0.5)^2 - (y - 0.5)^2 + 0.25 \right) \\
 &\quad e^{-10t} \cos \left(\pi \left(2(x - 0.5)^2 + 2(y - 0.5)^2 \right) \right), \\
 p(x, y, t) &= (\cos(4\pi x) + \cos(4\pi y)) e^{-10t}.
 \end{aligned} \tag{6.8}$$

In fig. 6.10, we plot the contour for velocity and pressure generated from eq. (6.8) for the concave domain. We note that the MS described remains the same for the corresponding packed version of the domains.

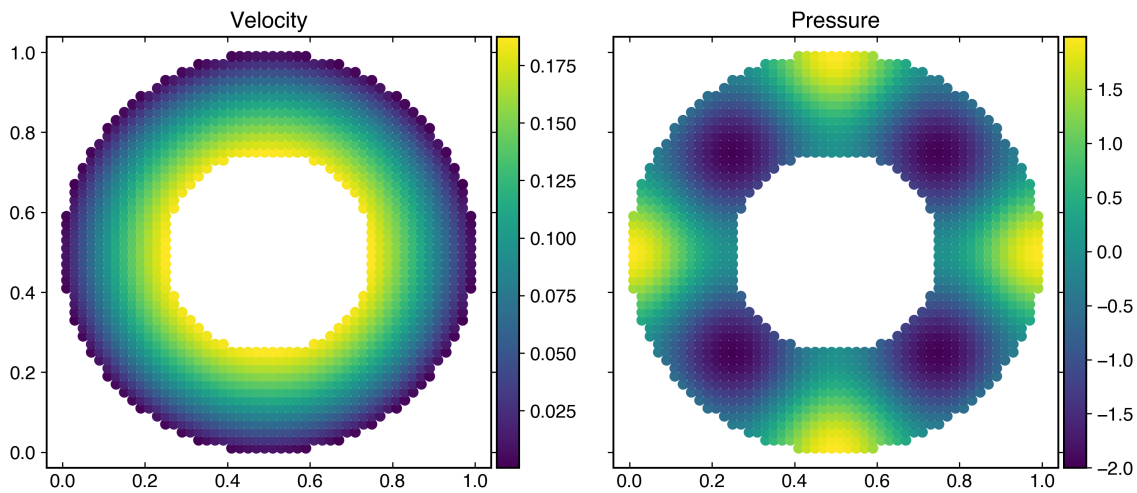


Figure 6.10 : Velocity and pressure contours on the concave domain of the MS in eq. (6.8).

6.1.4 Inlet and outlet velocity boundary condition

At the inlet, we make sure that $\nabla \mathbf{u} \cdot \hat{\mathbf{n}} = 0$. Since the inlet is usually straight. We consider one type of inlet with constant normal $\hat{\mathbf{n}} = -\mathbf{i}$ similarly outlet with normal $\hat{\mathbf{n}} = \mathbf{i}$. We use the MS given by

$$\begin{aligned} u(x, y, t) &= y(y-1)e^{-10t} \cos(2\pi y) + 1, \\ v(x, y, t) &= -x^2(x-1)^2 e^{-10t} \sin(2\pi y), \\ p(x, y, t) &= (\cos(4\pi x) + \cos(4\pi y)) e^{-10t}. \end{aligned} \quad (6.9)$$

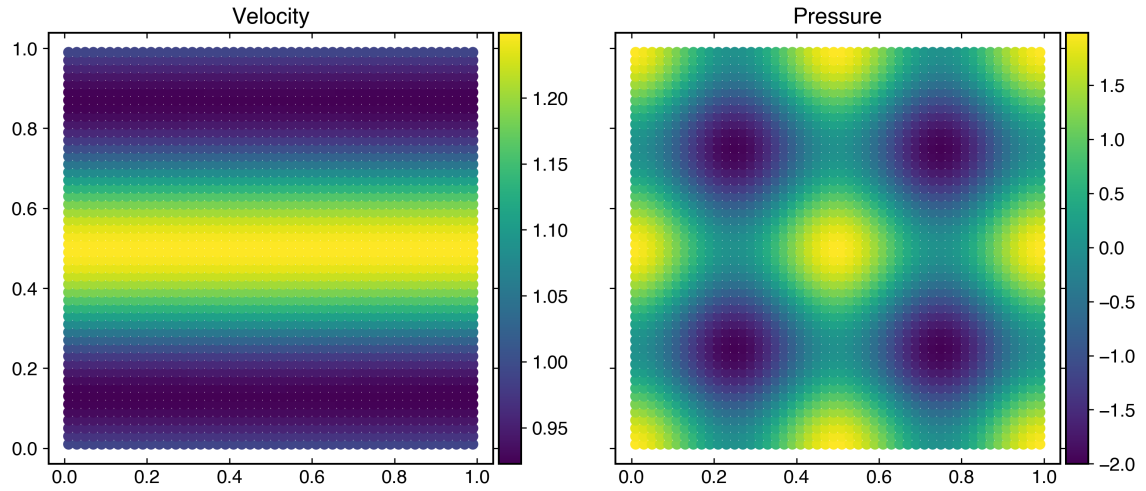


Figure 6.11 : Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.9).

In fig. 6.11, we plot the velocity and pressure contour for the MS in eq. (6.9). Additionally, we also simulate the wave passing through the inlet and outlet. We also must satisfy the boundary condition. For the inlet, we construct the MS given by

$$\begin{aligned} u(x, y, t) &= x^2 y(y-1) e^{-200(x-0.1-40t)^2} \cos(2\pi y) + 1, \\ v(x, y, t) &= 0.0, \\ p(x, y, t) &= \cos(4\pi x) + \cos(4\pi y). \end{aligned} \quad (6.10)$$

In fig. 6.12, we plot the velocity and pressure contour for the MS in eq. (6.10). We construct the wave of velocity passing through the outlet given by

$$\begin{aligned} u(x, y, t) &= (x-1)^2 y(y-1) e^{-200(x-0.9+40t)^2} \cos(2\pi y) + 1, \\ v(x, y, t) &= 0.0, \\ p(x, y, t) &= \cos(4\pi x) + \cos(4\pi y). \end{aligned} \quad (6.11)$$

In fig. 6.13, we plot the velocity and pressure generated by the MS in eq. (6.11).

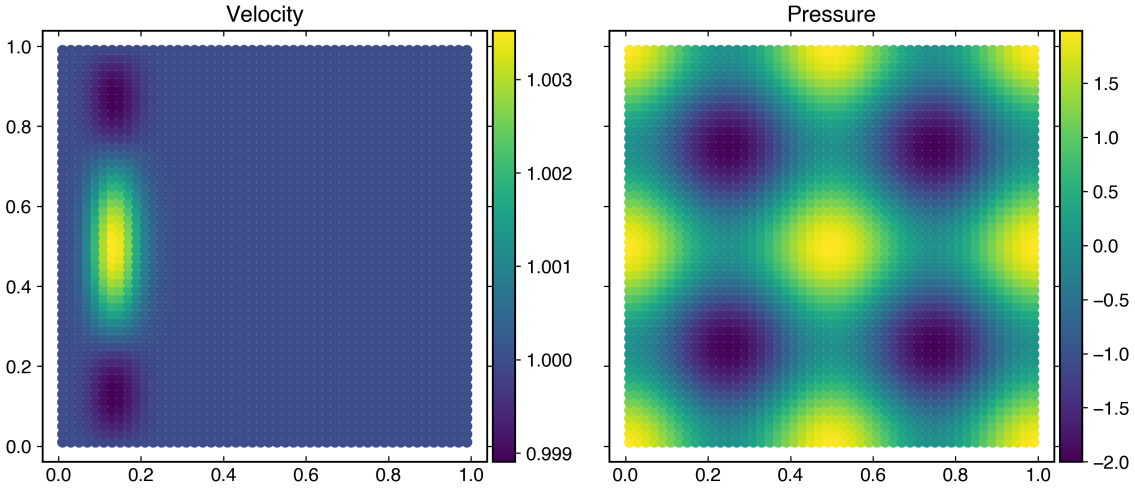


Figure 6.12 : Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.10).

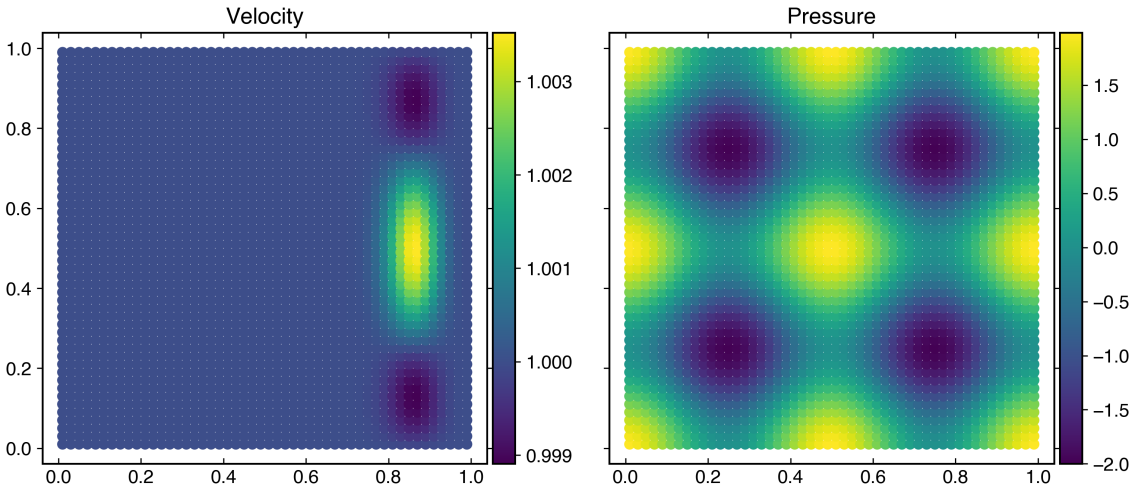


Figure 6.13 : Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.11).

6.1.5 Inlet and outlet pressure boundary condition

At the inlet, for pressure, we make sure that $\nabla p \cdot \hat{\mathbf{n}} = 0$. For the inlet as well as the outlet, we use the MS given by

$$\begin{aligned} u(x, y, t) &= y(y-1)e^{-10t} \cos(2\pi y) + 1, \\ v(x, y, t) &= -x(x-1)e^{-10t} \sin(2\pi y), \\ p(x, y, t) &= y(y-1)e^{-10t} \cos(2\pi y). \end{aligned} \quad (6.12)$$

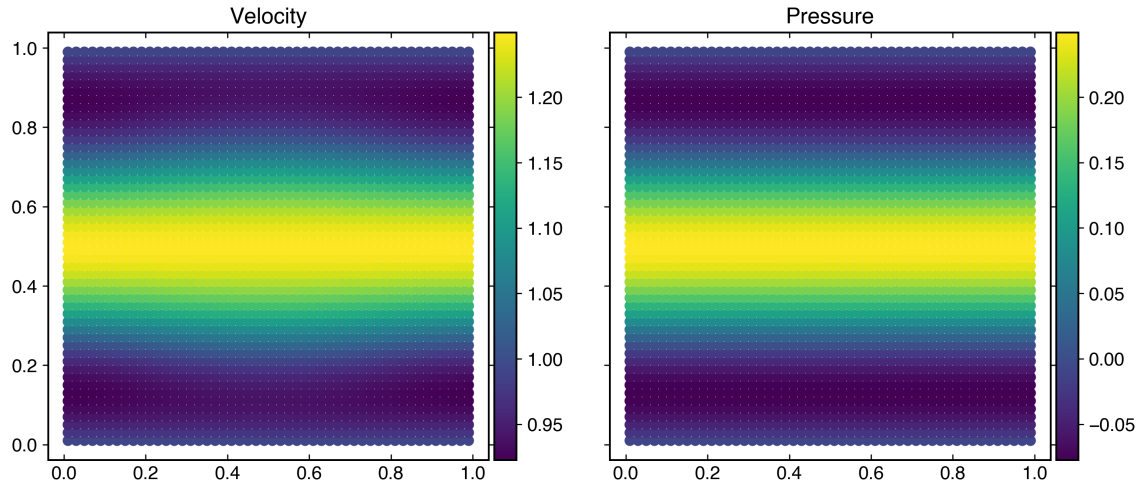


Figure 6.14 : Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.12).

In fig. 6.14, we plot the velocity and pressure contour generated from the MS in eq. (6.12). In order to simulate a pressure wave passing through both inlet and outlet, we construct MSs with pressure moving with the artificial speed of sound. For the inlet, we construct the MS given by

$$\begin{aligned} u(x, y, t) &= y(y-1) \cos(2\pi y) + 1, \\ v(x, y, t) &= 0.0, \\ p(x, y, t) &= x^2 e^{-200(x-0.1-40t)^2} \cos(2\pi y). \end{aligned} \quad (6.13)$$

In fig. 6.15, we plot the velocity and pressure generated from the MS in eq. (6.13). In the case of the outlet, we construct the MS given by

$$\begin{aligned} u(x, y, t) &= y(y-1) \cos(2\pi y) + 1, \\ v(x, y, t) &= 0.0, \\ p(x, y, t) &= (x-1)^2 e^{-200(x-0.9+40t)^2} \cos(2\pi y). \end{aligned} \quad (6.14)$$

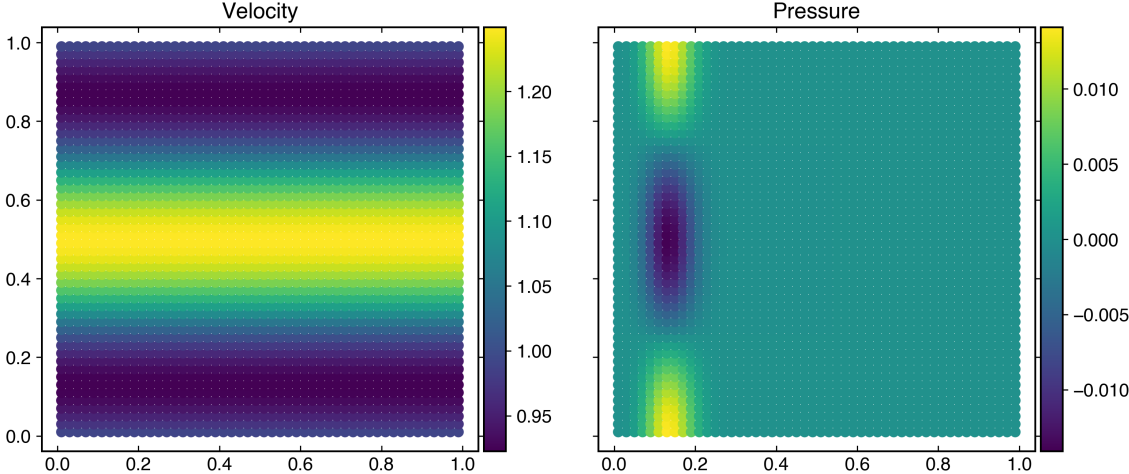


Figure 6.15 : Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.13).

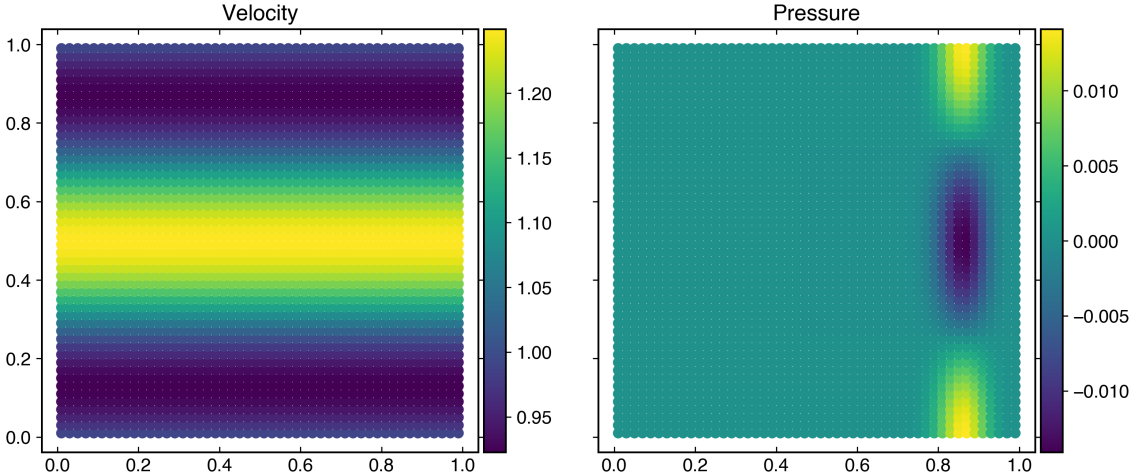


Figure 6.16 : Velocity and pressure contours on the domain in fig. 6.3 of the MS in eq. (6.14).

In fig. 6.16, we plot the velocity and pressure due to MS in eq. (6.14). In the next section, we use the solutions manufactured in this section to verify various boundary condition implementations.

6.2 Verification of BC implementations

In this section, we verify the convergence of various boundary condition implementations discussed in chapter 4 using MMS. We first show the convergence of various solid boundary condition implementations followed by the inflow and outflow boundary implementations. For all the test cases, we simulate 100 timesteps for resolutions in the range 100×100 to 500×500 , and evaluate the L_1 error using eq. (2.42). We set the time step corresponding to the highest resolutions as done in section 3.3.

We use the PySPH (Ramachandran et al., 2021) framework to implement all the methods. All the results presented in this work are reproducible and can be easily generated using the automation framework *automan* (Ramachandran, 2018). In the interest of reproducibility, we provide the entire source code at https://gitlab.com/pypr/mms_sph_bc.

6.2.1 Comparison of solid BC implementations

In this section, we verify all the solid boundary methods discussed in section 5.1.

Neumann pressure boundary condition

In this section, we apply various methods described in section 5.1 to apply the Neumann pressure boundary condition on the orange particles shown in the domains in fig. 6.1, and fig. 6.2. We first use the MS in eq. (6.2). The MS ensures that $\nabla p \cdot \hat{\mathbf{n}} = 0$ at the boundary of interest in the straight domain. We refer to a particular method using the corresponding first author names. The ‘MMS’ and ‘MMS-2L’ are the cases where properties on the solid are updated using the MS, and six layers and two layers of ghost particles are used to represent the solid, respectively. In fig. 6.17, we plot the L_1 error in the pressure and velocity after 100 timesteps.

The ‘MMS-2L’ plot shows that the L-IPST-C scheme converges even when two layers of solid particles are employed for a straight boundary. In the case of the straight domain, all the methods considered in this work are second-order convergent except the method proposed by Fourtakas et al. (2019), where a virtual stencil is used to complete the support of the particles. However, the rate of convergence is similar to as reported in Fourtakas et al. (2019).

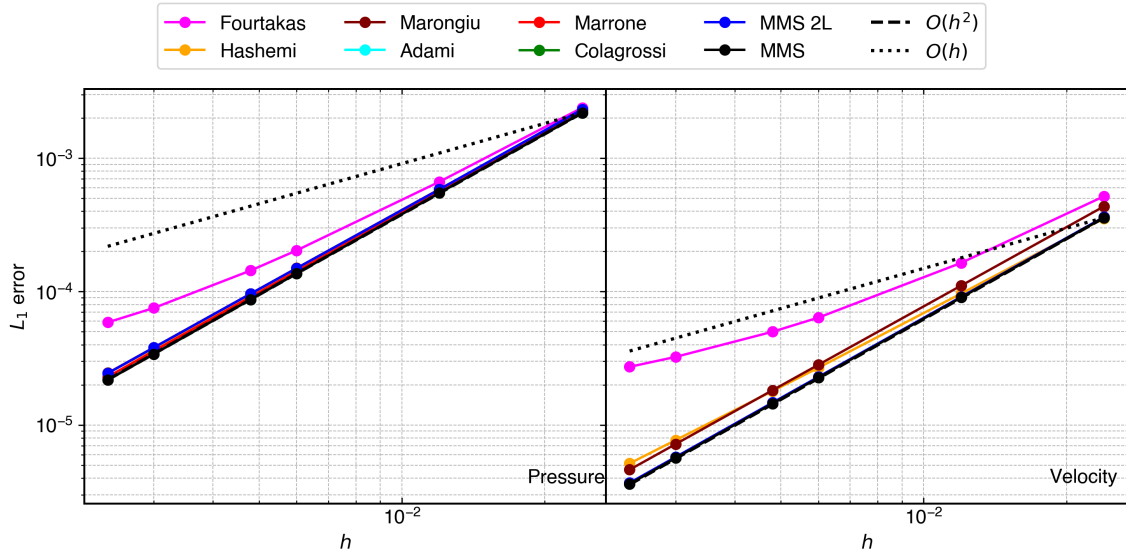


Figure 6.17 : L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the straight domain in fig. 6.1.

We also test all the methods on the convex and concave domains. In order to verify, we use the MS in eq. (6.3) which satisfies the boundary condition for the surface of interest in both domains. In fig. 6.18, and fig. 6.19, we plot the L_1 error for pressure and velocity for the convex and concave domains, respectively.

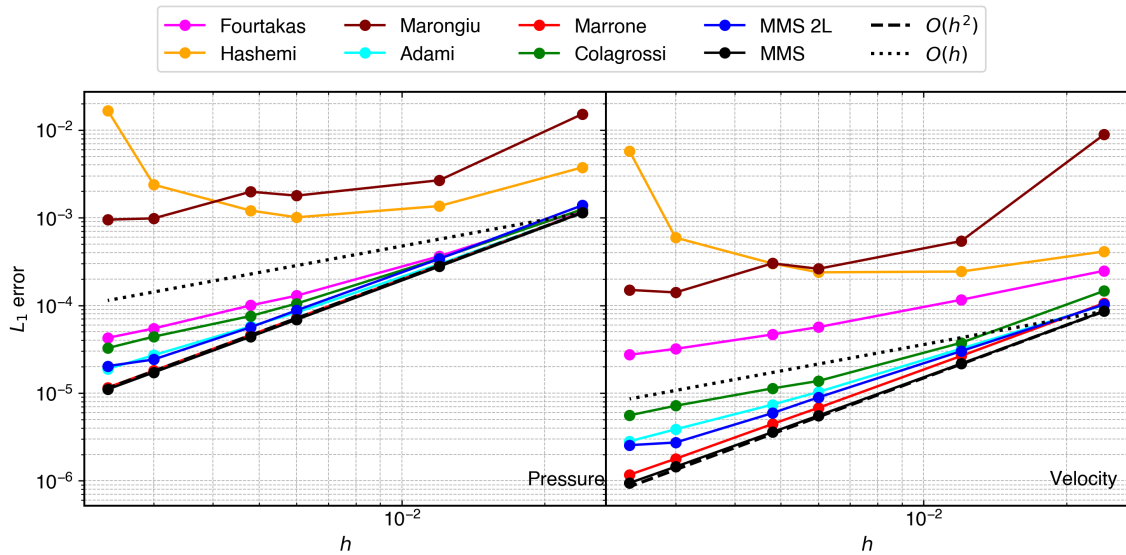


Figure 6.18 : L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the convex domain in fig. 6.1.

We observe that the method proposed by Hashemi et al. (2012) diverges since only a single layer of particles are used to represent the solid, which is insufficient even for corrected gradient computation. However, the method proposed by Marongiu et al. (2007)

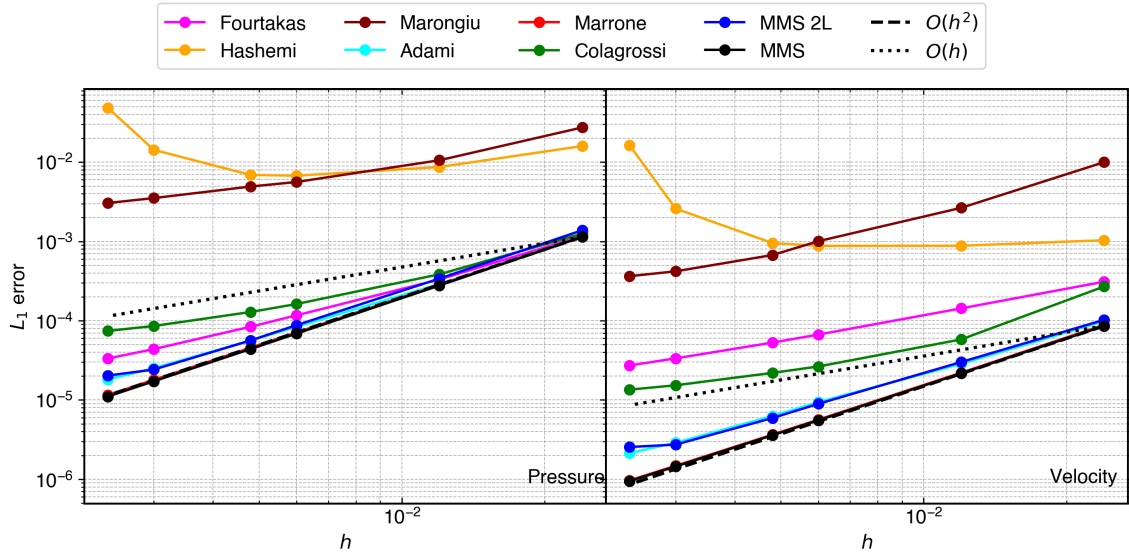


Figure 6.19 : L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the concave domain in fig. 6.1.

shows first-order convergence. Moreover, the rate of convergence is non-monotonous for a convex boundary. The reason behind a lower order of convergence is the use of a single layer of particles and zeroth-order interpolation on the virtual particles, which are further used in the fifth-order finite difference interpolation. The method proposed by Fourtakas et al. (2019) shows first-order convergence in pressure and 1.5 in velocity, as expected. The convergence of the method proposed by Adami et al. (2012), and ‘MMS-2L’ are very close to 1.5. The ‘MMS-2L’ has a slight decrease in convergence compared to ‘MMS’, which shows that the minimum number of layers required for an accurate Neumann boundary is higher for a curved surface compared to a straight boundary. Clearly, the method proposed by Marrone et al. (2011) is second-order convergent.

In order to remove the effect of jagged edges on the convergence of the boundary condition implementations, we performed the numerical experiment on the packed domain viz. packed-convex and packed concave as shown in fig. 6.2. Since the boundary surfaces are the same, we use the same MS in eq. (6.3). In fig. 6.20, and fig. 6.21, we plot the L_1 error for pressure and velocity for both domains.

We observe that all the methods show a better rate of convergence. We note that unlike earlier in the Hashemi method, errors do not increase. Furthermore, the Marongiu method shows an almost constant rate of convergence for a convex domain. The rate of convergence increases for all the methods compared to an unpacked domain. The convergence of the method proposed by Colagrossi et al. (2003) increases by a large amount since the particles after mirroring have good distribution. The method by Marrone

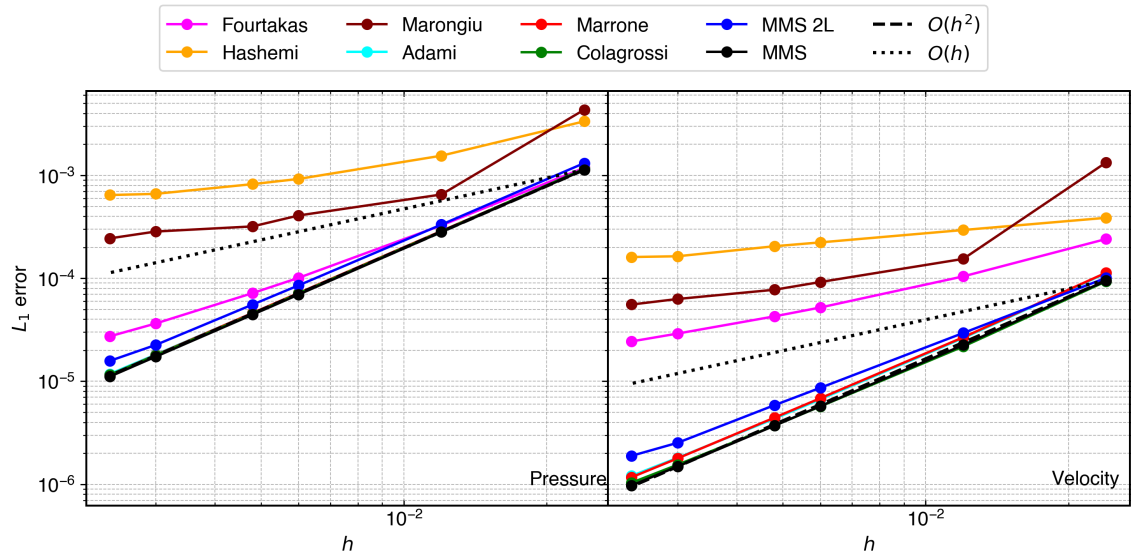


Figure 6.20 : L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the packed-convex domain in fig. 6.2.

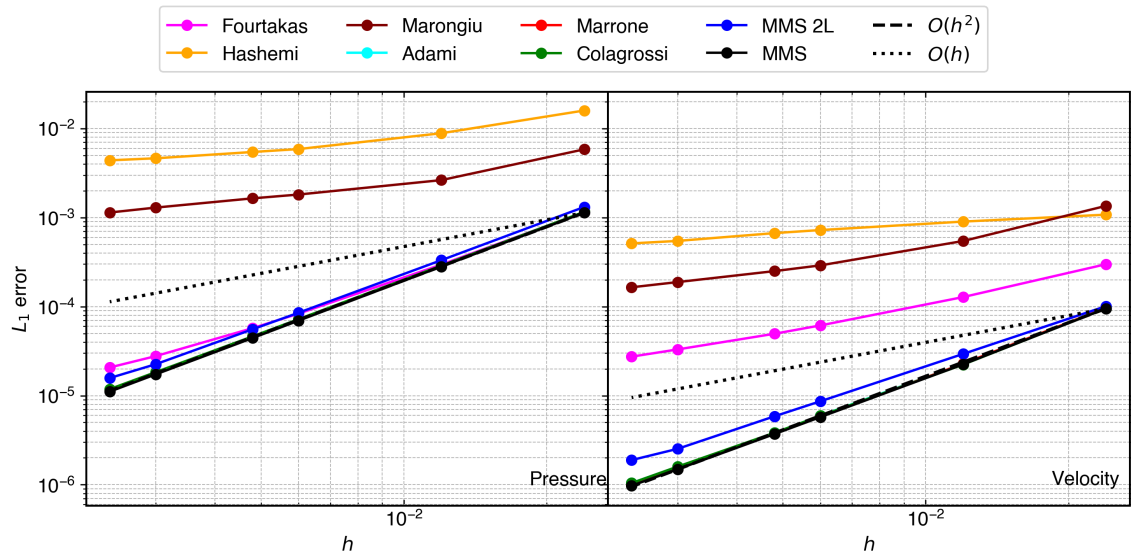


Figure 6.21 : L_1 error in pressure and velocity after 100 time steps different Neumann pressure boundary implementations in the packed-concave domain in fig. 6.2.

et al. (2011) and Colagrossi et al. (2003) overlaps and shows second-order convergence. This test also demonstrates the effectiveness of packing for curved surfaces.

Slip boundary condition

In this section, we test various slip boundary condition implementations discussed in section 5.1. In order to test these methods, we use all the different domains considered in the previous results. For the straight domain, we use the MS in eq. (6.4). In order to construct this MS, we ensure that $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$ at the boundary. In fig. 6.22, we plot the L_1 error in pressure and velocity after 100 timesteps. Clearly, all the methods show second-order convergence. In general, the slip boundary condition is not a realistic boundary condition, and it is usually used to remove the effect of walls not affecting the flow. However, to complete the discussion, we test these methods in other domains.

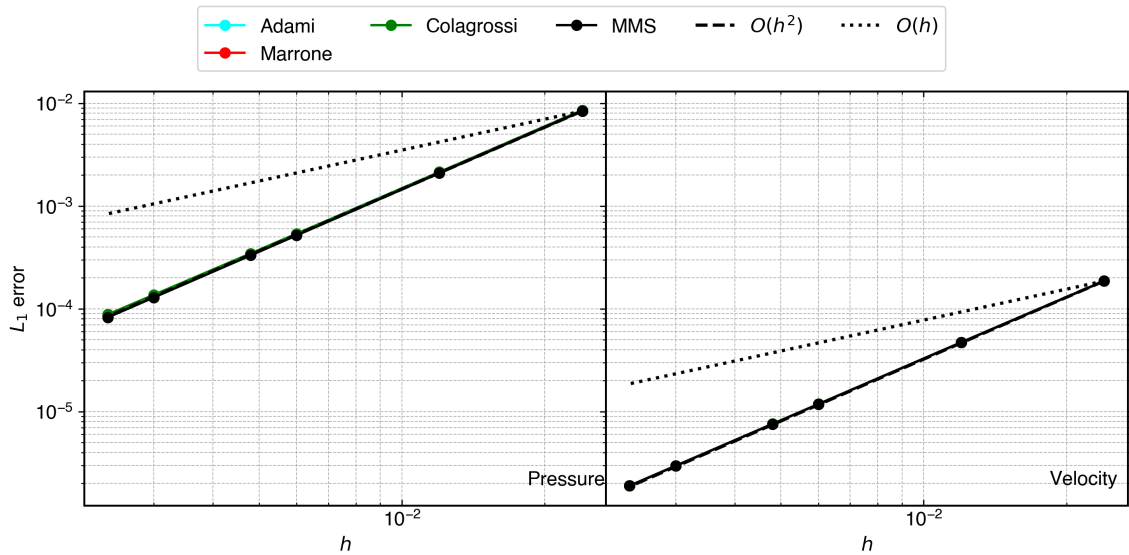


Figure 6.22: L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in straight domain in fig. 6.1.

We construct the MS for convex and concave domains in eq. (6.5) that satisfies $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$ at respective boundary surfaces of interest. In fig. 6.23 and fig. 6.24, we plot the L_1 error for pressure and velocity in convex and concave domains after 100 timesteps, respectively. Clearly, the method proposed by Colagrossi et al. (2003) diverges for higher resolutions. The method proposed by Adami et al. (2012) shows convergence rate close to 1.6 whereas the method by Marrone et al. (2011) is very close to second-order convergence.

We use the same MS for the packed-convex and packed-concave domains. In fig. 6.25 and fig. 6.26, we plot the L_1 error for pressure and velocity for packed-convex and packed-concave domains after 100 timesteps, respectively. As expected, The order

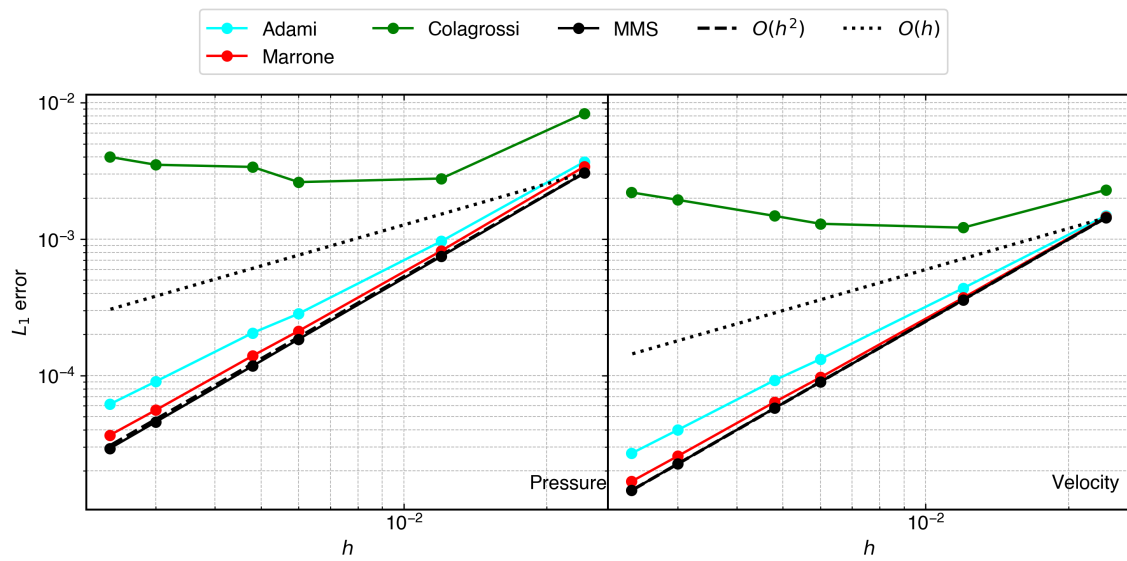


Figure 6.23 : L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the convex domain in fig. 6.1.

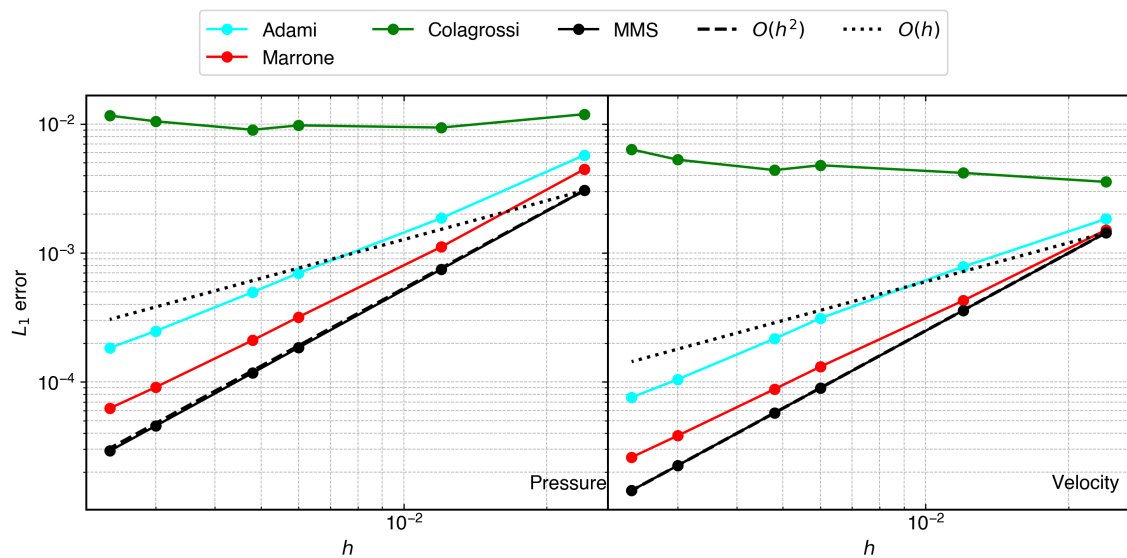


Figure 6.24 : L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the concave domain in fig. 6.1.

of convergence is improved. In the packed domain, the convergence for the method by Adami et al. (2012), and Marrone et al. (2011) shows second-order convergence. The method proposed by Colagrossi et al. (2003) converges for lower resolutions in the case of the packed-convex domain but diverges in the case of the packed-concave domain. This shows that mirroring the fluid particles for a curved surface does not result in a convergent boundary condition.

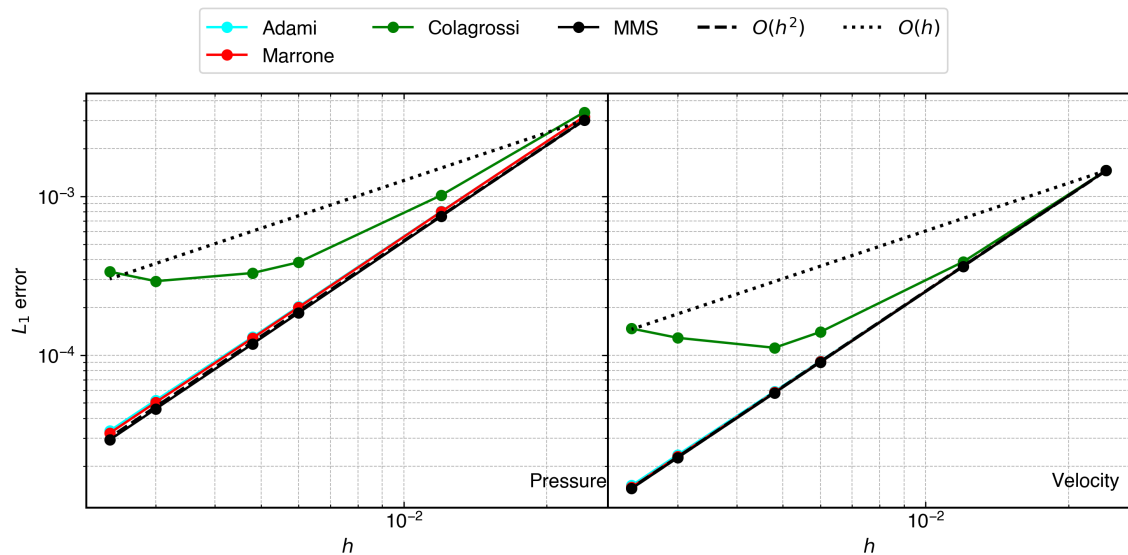


Figure 6.25 : L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the packed-convex domain in fig. 6.2.

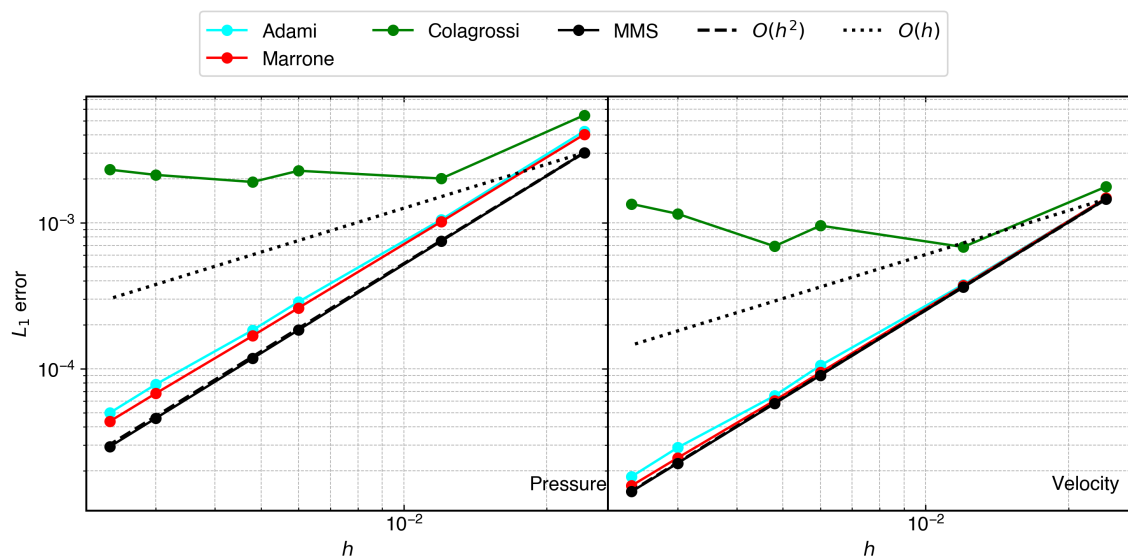


Figure 6.26 : L_1 error in pressure and velocity after 100 time steps for different slip boundary implementations in the packed-concave domain in fig. 6.2.

No-slip boundary condition

In this section, we test different no-slip boundary implementations discussed in section 5.1 using the domains used in the previous section. In all the no-slip boundary condition implementations, we apply no-penetration along with the no-slip boundary. In order to construct an MS for no-slip boundary condition, we satisfy $\mathbf{u} = 0$ at the boundary. For the straight domain, we use the MS in eq. (6.6). In fig. 6.27, we plot the L_1 error for pressure and velocity in the domain after 100 timesteps. Clearly, all the methods show a convergence rate very close to second-order.

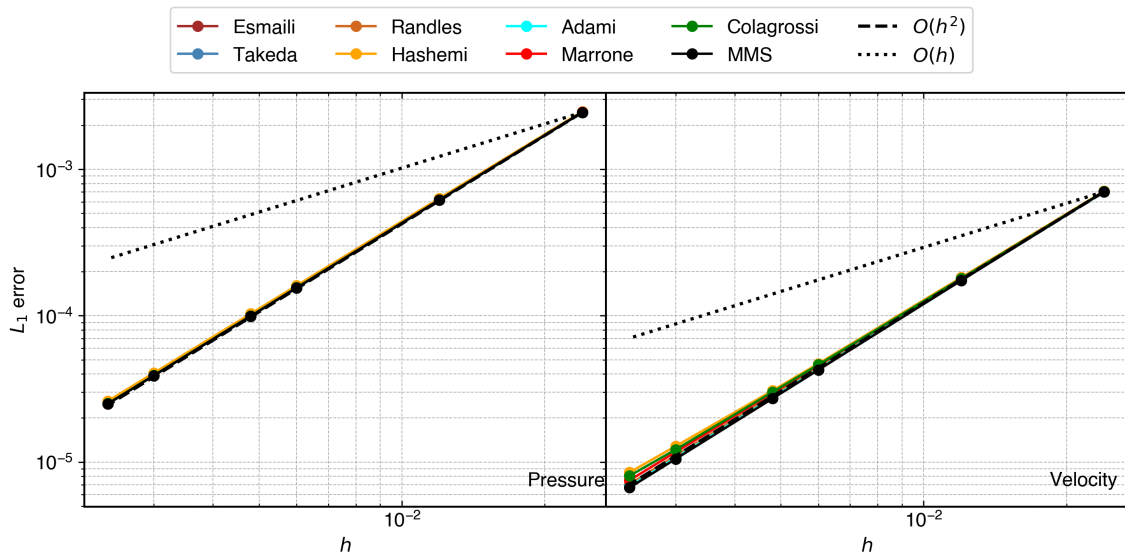


Figure 6.27 : L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the straight domain in fig. 6.1.

Generally, we find that objects on which we intend to apply no-slip boundary are curved. Therefore, we simulate all the methods on the domains having curved surfaces. For the convex domain, we use the MS in eq. (6.7) whereas, for the concave domain, we use eq. (6.8). In fig. 6.28 and fig. 6.29, we plot the L_1 error in pressure and velocity for convex and concave domains, respectively. Clearly, the method proposed by Hashemi et al. (2012) diverges for a curved surface. The errors in the solutions are more in the concave domain compared to the convex domain. The method by Randles et al. (1996), Esmaili Sikarudi et al. (2016), and Adami et al. (2012) shows first-order convergence. Whereas methods by Colagrossi et al. (2003) and Marrone et al. (2011) show close to 1.5. Some methods like Takeda et al. (1994) cannot be applied on the jagged boundary as some particles may lie on the surface, which may result in zero in the denominator of eq. (5.7).

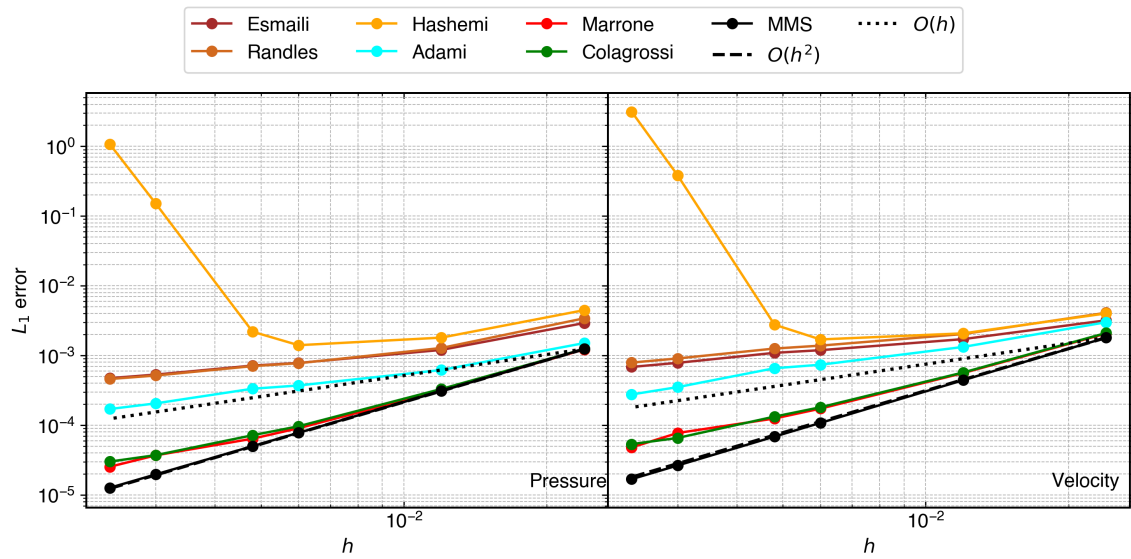


Figure 6.28 : L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the convex domain in fig. 6.1.

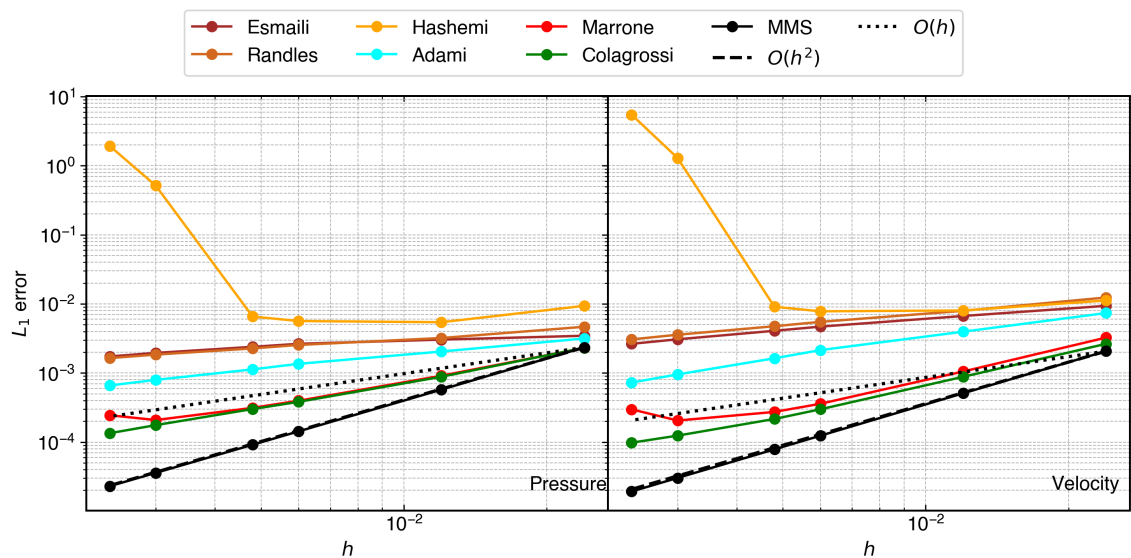


Figure 6.29 : L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the concave domain in fig. 6.1.

Similar to other tests, we use the packed version of convex and concave domains to test the convergence of the methods on packed domains. We use the MS in eq. (6.7), and eq. (6.8) for the packed-convex and packed-concave domains, respectively. In the figure fig. 6.30, and fig. 6.31, we plot the L_1 error in pressure and velocity for packed-convex and packed-concave domains, respectively. As expected, the convergence is improved. The method by Adami et al. (2012) does not show any convergence due to zero-order interpolation used on the ghost particles. Method by Takeda et al. (1994), Hashemi et al. (2012), Randles et al. (1996), and Esmaili Sikarudi et al. (2016) shows close to first-order convergence. Clearly, the method by Marrone et al. (2011) shows convergence close to the method when MS is used on the ghost particles. Further, the method of Colagrossi et al. (2003) also shows good convergence; however, the error compared to Marrone et al. (2011) method is 2 order of magnitude higher. In the case of the packed-concave domain in fig. 6.31, the order of convergence shown by all methods is lower compared to the packed-convex domain results.

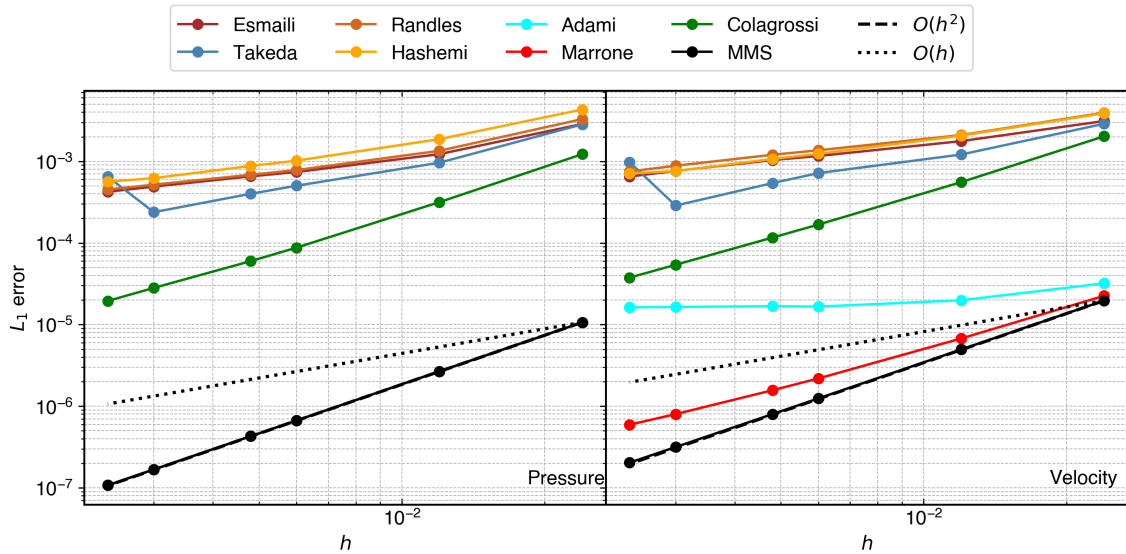


Figure 6.30 : L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the packed-convex domain in fig. 6.2. Note that the Marrone plot overlaps the plot of MMS.

In order to summarize the results, since the straight and convex domain shows better results compared to the concave domain, we consider the results for a concave domain only. Furthermore, we compile results for a packed domain only since the packed domains are preferred over the unpacked ones. In the case of the no-slip and slip boundary, we focus on the convergence of velocity, and in the case of the Neumann pressure, we focus only on the convergence of pressure. In table 6.1, we tabulate the error at the highest resolution, i.e., $\Delta x = 1/500$, and the approximate order of convergence for all the

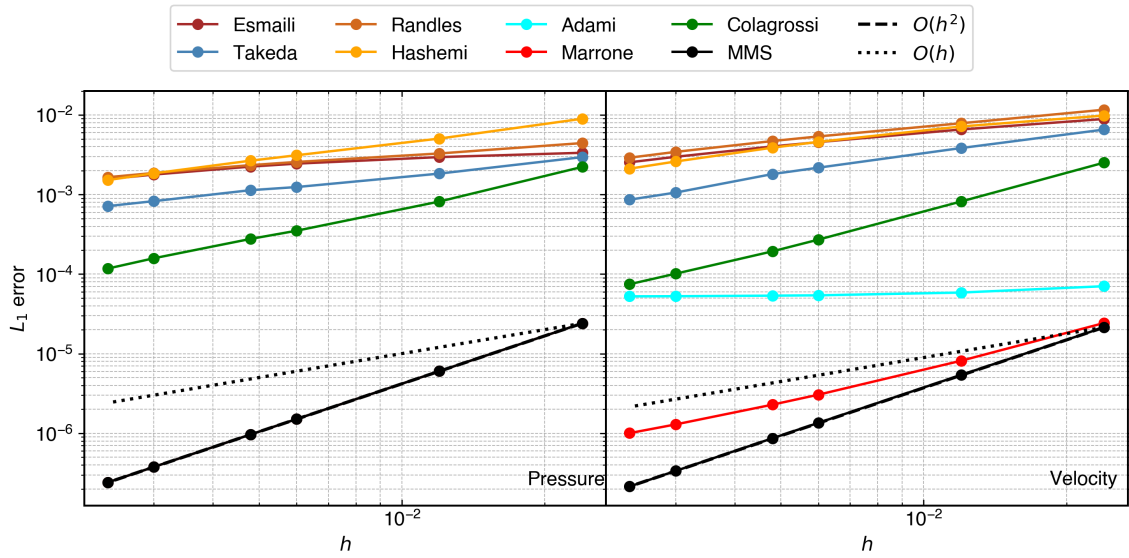


Figure 6.31 : L_1 error in pressure and velocity after 100 time steps for different no-slip boundary implementations in the packed-concave domain in fig. 6.2. Note that Marrone results overlaps the results of MMS.

Method	Neumann Pressure	Slip	No-Slip
Adami	$(1.17 \times 10^{-5})2.00$	$(1.82 \times 10^{-5})1.95$	$(5.26 \times 10^{-5})0.09$
Colagrossi	$(1.18 \times 10^{-5})2.00$	$(1.34 \times 10^{-3})0.19$	$(7.47 \times 10^{-5})1.52$
Esmaili	-	-	$(2.54 \times 10^{-3})0.55$
Fourtakas	$(2.07 \times 10^{-5})1.66$	-	-
Hashemi	$(4.39 \times 10^{-3})0.47$	-	$(2.11 \times 10^{-3})0.72$
Marongiu	$(1.14 \times 10^{-3})0.63$	-	-
Marrone	$(1.15 \times 10^{-5})2.00$	$(1.58 \times 10^{-5})1.97$	$(1.00 \times 10^{-6})1.35$
Randles	-	-	$(2.91 \times 10^{-3})0.62$
Takeda	-	-	$(8.63 \times 10^{-4})0.86$

Table 6.1 : Table showing the summary of the error (in brackets) at the resolution 500×500 and order of convergence of various boundary condition methods in the packed-concave domain.

boundary conditions and methods. Clearly, in the case of the Neumann pressure boundary condition, Adami, Colagrossi, and Marrone converges well. In the case of the slip boundary condition, only the Adami and Marrone methods work. Whereas in the case of the no-slip boundary, only Colagrossi and Marrone method show reasonable convergence. Clearly, the Marrone method is able to reach the lowest error as well as show convergence for all the types of boundary conditions.

6.2.2 Comparison of open BC implementations

In this section, we test various inlet and outlet boundary condition implementations discussed in section 5.2.

Inlet boundary

In order to test the inlet velocity boundary condition, we use the MS in eq. (6.9). In fig. 6.32, we plot the L_1 error in pressure and velocity after 100 timesteps for all the velocity inflow boundary implementations. We test all the methods discussed in section 5.2 viz. mirror, simple-mirror, and hybrid. We observe that both mirror and simple-mirror perform well for a velocity inlet boundary condition. Whereas the hybrid is bounded by the limiting error in both pressure and velocity. In the hybrid method, as discussed in section 5.2, the mean flow velocity of the inflow particles does not change and always prescribed. Moreover, any wave coming from the fluid is extrapolated in x -direction and does not affect the y -direction velocity causing a limiting error proportional to the speed of sound. An error estimation can be done in the future to determine the order of error terms.

In order to test the pressure inflow boundary implementation, we use the MS in eq. (6.12). In fig. 6.33, we plot the L_1 error in pressure and velocity after 100 timesteps for all the velocity inflow boundary implementations. Clearly, the boundary implementation for a pressure inflow boundary is second-order accurate for all the methods. In the case of the hybrid method, a slight deviation in the convergence can be seen.

In WCSPH schemes, due to weakly compressible assumption, waves travel with a speed of artificial velocity of sound. We use the MS to simulate a wave passing out of the inlet. These kinds of waves are encountered when a jump start is performed on a wind tunnel kind of simulation. We simulate the problem for 500 iterations in order to allow the wave to completely pass through the inlet/outlet. We use the MS in eq. (6.10) for the inlet velocity wave. In fig. 6.34, we plot the L_1 error in pressure and velocity for all the methods. Clearly, the hybrid method also shows second-order convergence along with other methods.

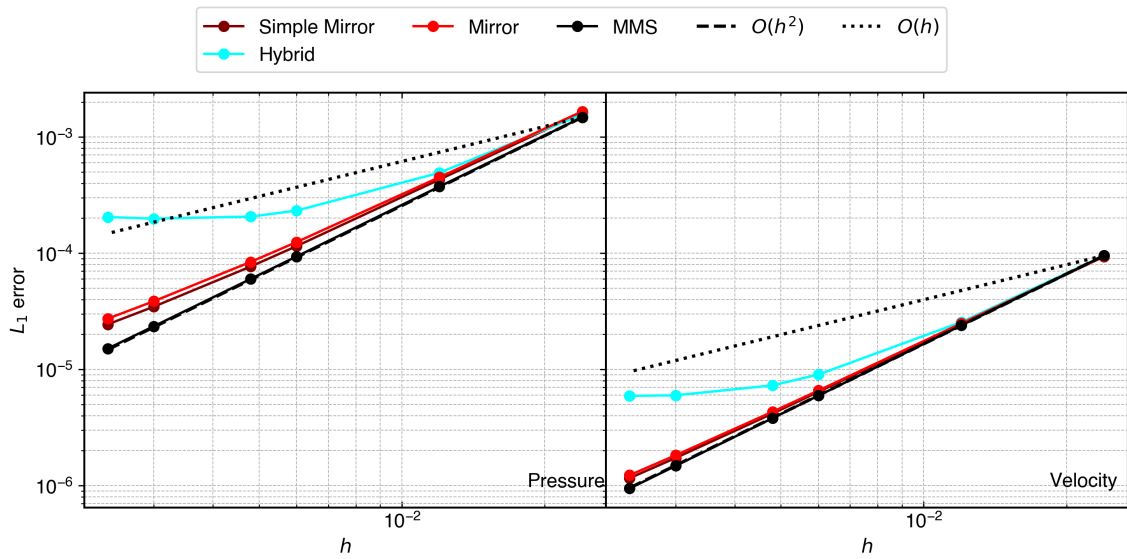


Figure 6.32 : L_1 error in pressure and velocity after 100 time steps for different inlet velocity boundary implementations in the domain in fig. 6.3.

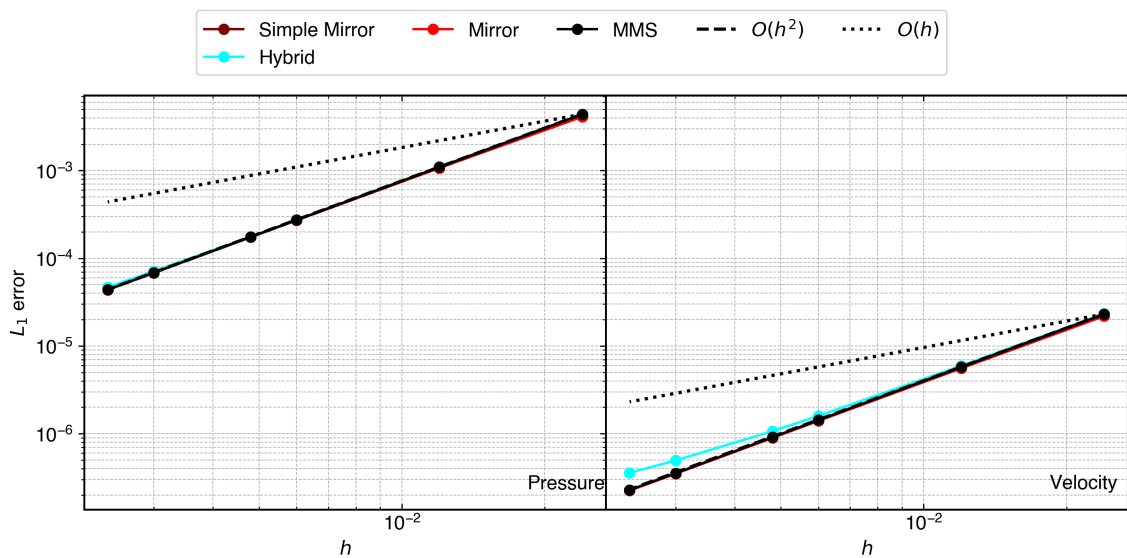


Figure 6.33 : L_1 error in pressure and velocity after 100 time steps for different inlet pressure boundary implementations in the domain in fig. 6.3.

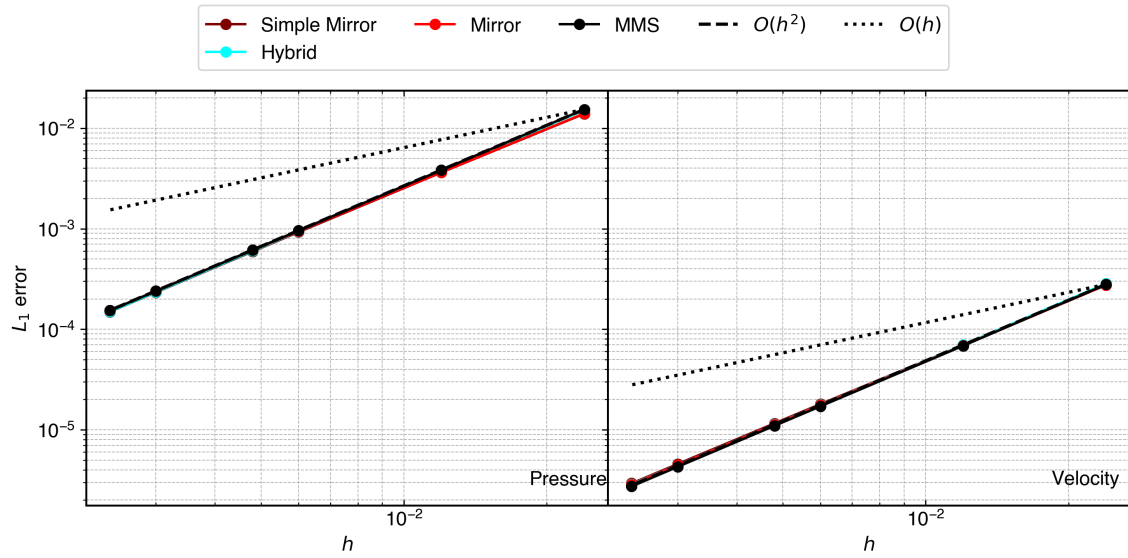


Figure 6.34 : L_1 error in pressure and velocity after 500 time steps for different inlet velocity wave going upstream boundary implementations in the domain in fig. 6.3.

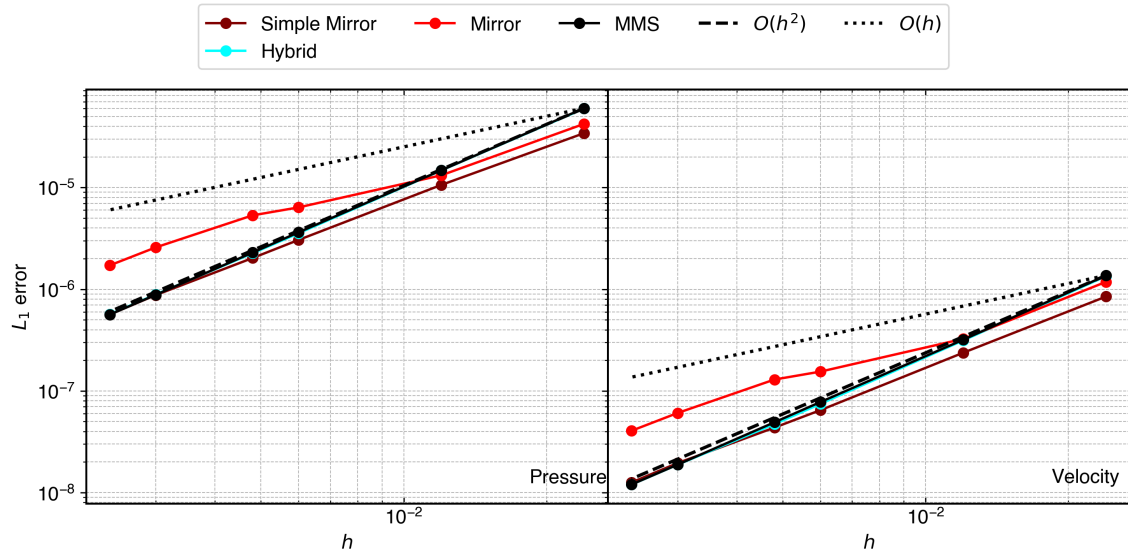


Figure 6.35 : L_1 error in pressure and velocity after 500 time steps for different inlet pressure wave going upstream boundary implementations in the domain in fig. 6.3.

In order to simulate a pressure wave going out of the inlet, we use the MS in eq. (6.13). In fig. 6.35, we plot the L_1 error in pressure and velocity for all the methods. The mirror method shows a slight increase in error for higher resolutions. The simple-mirror remains at the same level of error compared to the hybrid method, which shows second-order convergence.

Outlet boundary

The outflow is different compared to the inlet, as we usually do not have any information about the ghost particles in these regions. In order to test the outflow velocity boundary condition, we use the MS in eq. (6.9). In fig. 6.36, we plot the L_1 error in pressure and velocity after 100 timesteps for all the velocity outflow boundary implementations. The do-nothing and the hybrid boundary are both bounded by a limiting error which is proportional to the speed of sound. As before, both mirror and simple-mirror show second-order convergence for velocity outlet boundary condition.

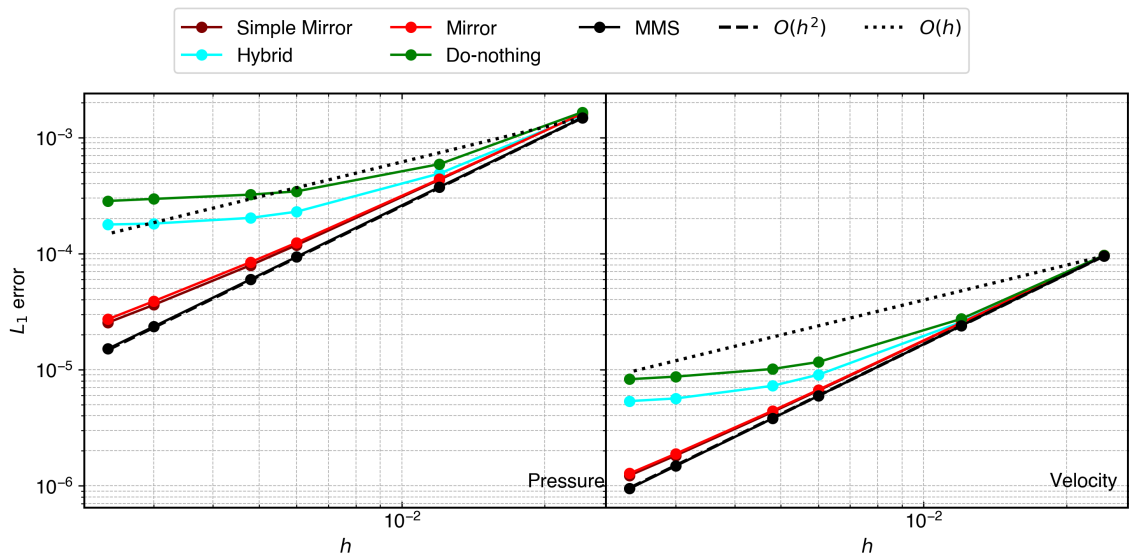


Figure 6.36: L_1 error in pressure and velocity after 100 time steps for different outlet velocity boundary implementations in the domain in fig. 6.3.

In order to test the pressure outflow boundary implementation, we use the MS in eq. (6.12). In fig. 6.37, we plot the L_1 error in pressure and velocity after 100 timesteps for all the velocity outflow boundary implementation. Clearly, all the methods show second-order convergence.

To investigate the behavior of the outlet boundary implementation under the influence of a passing wave, we use the MS in eq. (6.14) and eq. (6.11) for outlet pressure and outlet velocity boundary implementations, respectively. In fig. 6.38, and fig. 6.39, we plot

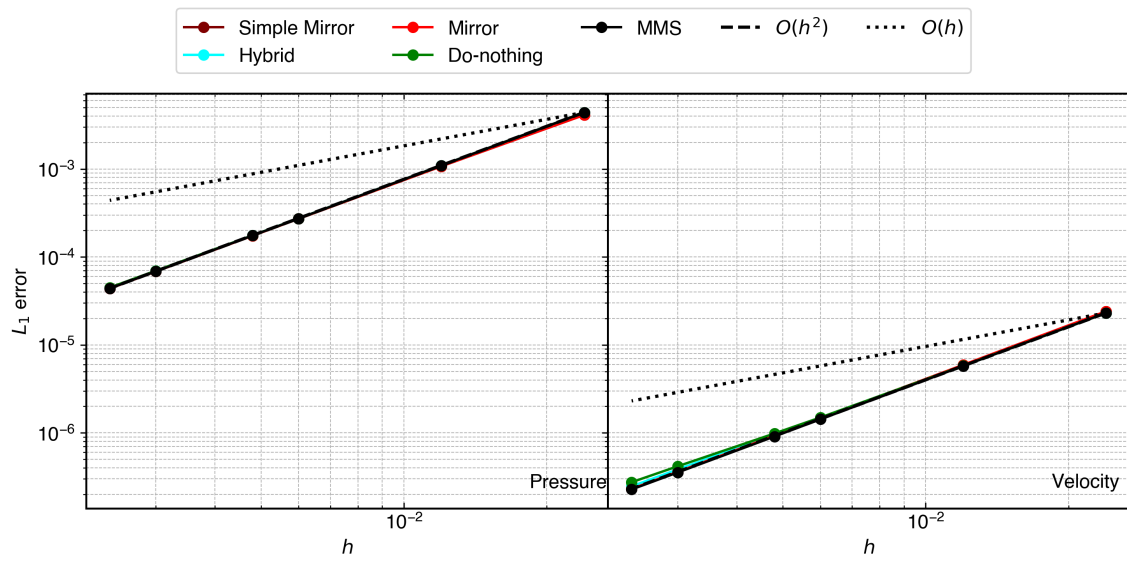


Figure 6.37 : L_1 error in pressure and velocity after 100 time steps for different outlet pressure boundary implementations in the domain in fig. 6.3.

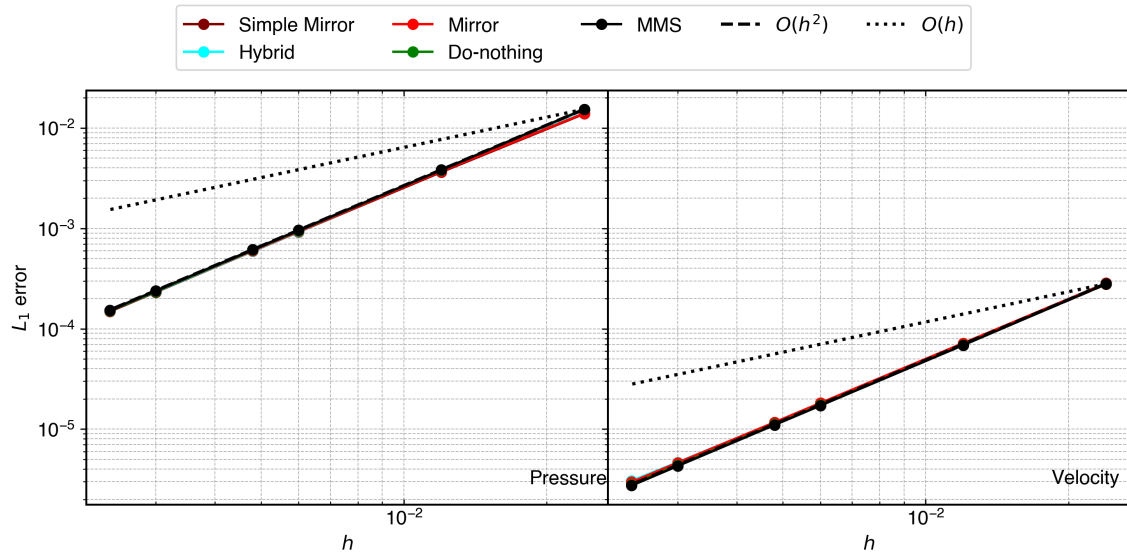


Figure 6.38 : L_1 error in pressure and velocity after 500 time steps for different outlet velocity wave going downstream boundary implementations in the domain in fig. 6.3.

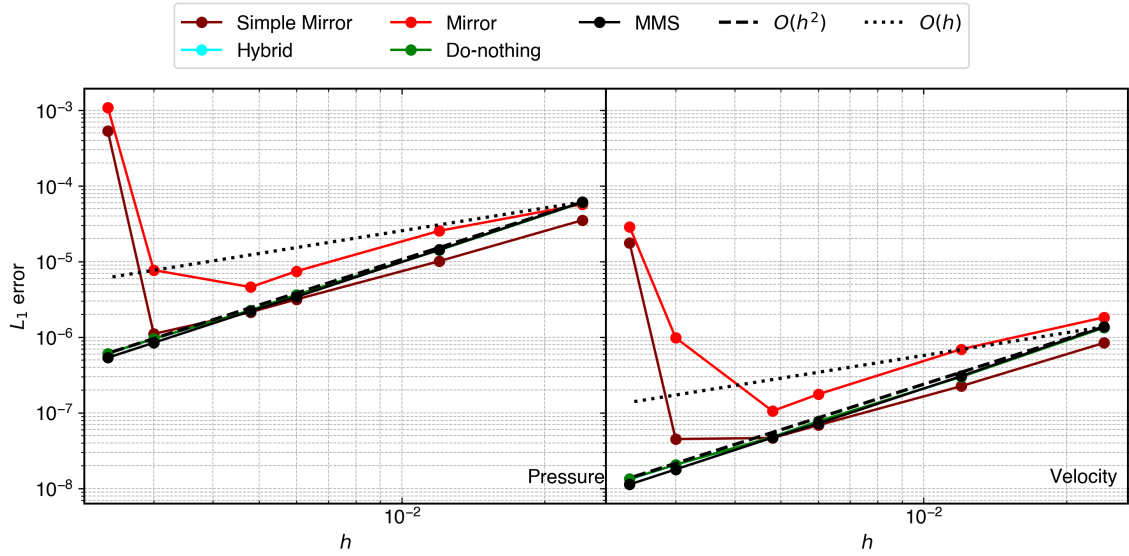


Figure 6.39 : L_1 error in pressure and velocity after 500 time steps for different outlet pressure boundary wave going downstream implementations in the domain in fig. 6.3.

Method	Velocity in	Pressure in
Hybrid	(2.87×10^{-6}) 2.00	(5.66×10^{-7}) 2.02
Mirror	(2.84×10^{-6}) 1.97	(1.71×10^{-6}) 1.38
Simple Mirror	(2.93×10^{-6}) 1.98	(5.64×10^{-7}) 1.81

Table 6.2 : Summary of results for the wave traveling upstream out of the inlet for all the methods. Error at the highest resolution is shown in brackets.

the L_1 error for pressure and velocity after 500 timesteps for both MSs. In the case of the velocity wave, all the methods show second-order convergence. However, in the case of the pressure wave in fig. 6.39, the mirror and simple-mirror method diverges. This shows that the mirror and simple-mirror methods are not truly non-reflecting and are unable to pass a pressure wave. These results support the finding in the previous chapter, where a short domain with the mirror outlet was found to be unstable.

In order to summarize the results for the open boundary conditions, we consider only the results for the traveling wave since, in WCSPH, it is important that the waves that are generated must be allowed to pass through the inlet/outlet without affecting the flow. In the case of a velocity wave, we focus on errors in velocity, whereas in the case of a pressure wave, we focus on errors in pressure. In table 6.2 and table 6.3, we tabulate the error for the highest resolution and the approximate order of convergence for all the methods simulating traveling wave MS. Clearly, the mirror method shows a significant decrease in order of convergence in the case of the pressure wave moving upstream. In

Method	Velocity out	Pressure out
Do-nothing	(2.80×10^{-6}) 2.00	(6.08×10^{-7}) 2.00
Hybrid	(3.05×10^{-6}) 1.99	(6.05×10^{-7}) 2.00
Mirror	(2.97×10^{-6}) 1.97	(1.01×10^{-3}) -3.63
Simple Mirror	(2.86×10^{-6}) 1.97	(5.34×10^{-4}) -4.21

Table 6.3 : Summary of results for the wave traveling downstream out of the outlet for all the methods. Error at the highest resolution is shown in brackets.

the case of the wave traveling downstream, both mirror and simple-mirror diverge. The hybrid method is applicable and converges for both scenarios.

6.3 Performance comparison

In this section, we compare the performance of three solid boundary condition implementations, viz. Marrone, Colagrossi, and Adami. For this test case, we use a 10-core, dual-socket Intel(R) Xeon(R) CPU E5-2650 v3 processor CPU. In the context of complexity, the Adami method requires only one loop to extrapolate properties from fluid, whereas the Colagrossi method requires the creation and deletion of particles in every timestep. In the case of the Marrone method, one needs to solve an additional 4×4 matrix for each particle in the solid boundary, excluding the extrapolation step.

In fig. 6.40, we plot the time taken versus the no of parallel computing threads for 100 timesteps for a 100×100 domain. Clearly, the time taken by the methods are very close despite the fact that different amount of computations are required. This is due to a very low number of solid particles compared to fluid particles. In the present case, the fluid particles are 10000, whereas the solid particles on which the boundary condition is implemented are 600. Therefore, we demonstrated that a higher-order boundary implementation does not affect the performance of the code. Furthermore, these results can be easily extrapolated to open-boundary implementations. In the next section, we propose an algorithm to obtain a convergent solver for a problem containing inlet, outlet, and solid boundaries.

6.4 Complete second-order convergent SPH scheme

In the previous section, we have shown that some of the boundary condition implementations are convergent using the MMS. In chapter 2 and chapter 3, we have used verification methods to procedurally obtain a second-order accurate WCSPH scheme

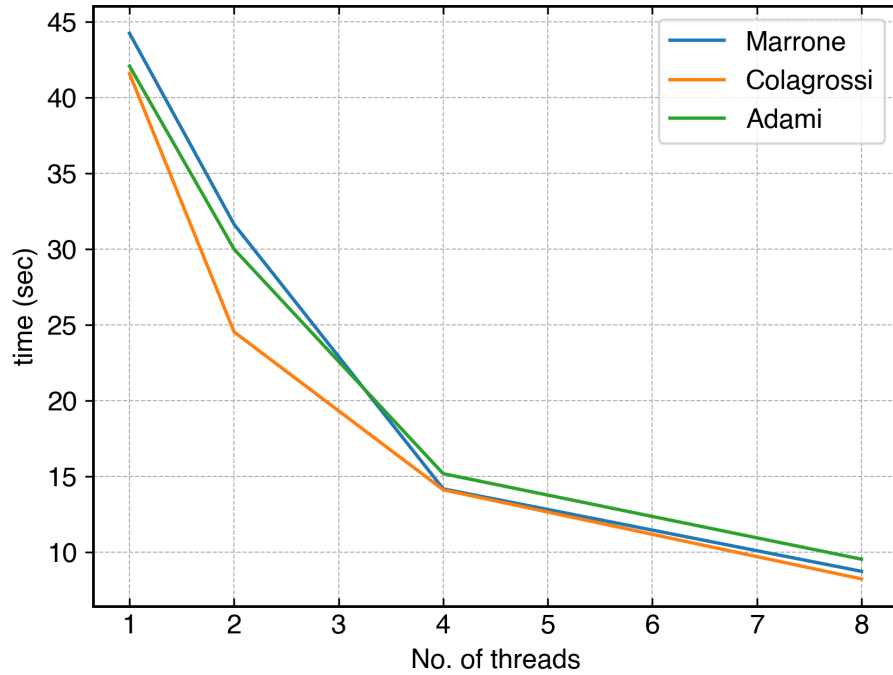


Figure 6.40 : The time taken with the increase in the number of threads for different solid boundary conditions.

without boundary. In this section, we extend our method to propose a second-order convergent scheme with second-order convergent boundary implementations. For brevity, we use short names to represent an equation in the algorithm. For example, **EvaluateVelocityOnGhost(dest, sources)** can be written formally as shown in algorithm 6.

Algorithm 6: Psuedo-code of equation *EvaluateVelocityOnGhost*.

```

for  $i$  in  $dest$  do
   $u_i = 0$ ;
  for  $j$  in  $sources$  do
     $u_i = u_i + u_j \omega_j W_{ij}$ ;

```

In algorithm 6, i is the loop index, and j is over all the neighbors of i^{th} element. We note that all the extrapolation/equation require a corrected kernel/gradient and is implied.

Consider a fluid domain, the inlet continuously feeding particles to the fluid, and the outlet continuously consuming particles from the fluid and an arbitrarily-shaped solid body. In algorithm 7, we show the algorithm for a convergent time-accurate WCSPH scheme to simulate the flow for a given initial condition. We denote all the fluid particles

by \mathcal{F} , all solid particles by \mathcal{S} , and all inlet/outlet particles by \mathcal{JO} . All the virtual particles required for solid particles are denoted by $\mathcal{M}(\mathcal{S})$.

Algorithm 7: A Second-order convergent scheme.

```

while  $t < t_{final}$  do
  for  $i$  in  $\mathcal{F}$  do
    EvaluatePressure(dest= $i$ , sources= $\phi$ );
  for  $i$  in  $\mathcal{JO}$  do
    EvaluateVelocityOnInletOutlet(dest= $i$ , sources= $\mathcal{F}$ );
    EvaluatePressureOnInletOutlet(dest= $i$ , sources= $\mathcal{F}$ );
  for  $i$  in  $\mathcal{M}(\mathcal{S})$  do
    EvaluateVelocityOnGhost(dest= $i$ , sources= $\mathcal{F} \cup \mathcal{JO}$ );
    EvaluatePressureOnGhost(dest= $i$ , sources= $\mathcal{F} \cup \mathcal{JO}$ );
  for  $i$  in  $\mathcal{S}$  do
    EvaluateSlipVelocityOnSolidFromGhost(dest= $i$ , sources= $\phi$ );
    EvaluateNoSlipVelocityOnSolidFromGhost(dest= $i$ , sources= $\phi$ );
    EvaluatePressureOnSolidFromGhost(dest= $i$ , sources= $\phi$ );
  for  $i$  in  $\mathcal{F} \cup \mathcal{S}$  do
    ComputeVelocityGradient(dest= $i$ , sources= $\mathcal{F} \cup \mathcal{JO}$ );
    # use extrapolated no slip velocity for solid
    ComputeVelocityGradientSolid(dest= $i$ , sources= $\mathcal{S}$ );
  for  $i$  in  $\mathcal{F}$  do
    ContinuityEquation(dest= $i$ , sources= $\mathcal{F} \cup \mathcal{JO}$ );
    # use extrapolated slip velocity for solid
    ContinuityEquationSolid(dest= $i$ , sources= $\mathcal{S}$ );
    PressureForces(dest= $i$ , sources= $\mathcal{F} \cup \mathcal{S} \cup \mathcal{JO}$ );
    ComputeViscousForces(dest= $i$ , sources= $\mathcal{F} \cup \mathcal{JO}$ );
  for  $i$  in  $\mathcal{F} \cup \mathcal{JO}$  do
    Integrate(dest= $i$ , sources= $\phi$ )

```

The algorithm starts with the evaluation of pressure from the equation of state given by eq. (2.27) in **EvaluatePressure**. In the next step, we evaluate pressure and velocity on the inlet and outlet regions using the hybrid method in section 4.1.3 in **EvaluatePressureOnInletOutlet**, and **EvaluateVelocityOnInletOutlet**, respectively. We note that the inlet and outlet properties are updated, and then these are used as the

source for solid properties in case of overlaps. We use the method by Marrone et al. (2011) to evaluate the properties of solid particles. We first evaluate accurate first-order values on virtual particles in **EvaluateVelocityOnGhost** and **EvaluatePressureOnGhost**, and then use these values to implement pressure, slip, and no-slip boundary conditions in **EvaluatePressureOnSolidFromGhost**, **EvaluateSlipVelocityOnSolidFromGhost**, **EvaluateNoSlipVelocityOnSolidFromGhost**, respectively. In order to obtain accurate second-order viscous operator, we require the velocity gradient on each particle, which is computed in **ComputeVelocityGradient**, and **ComputeVelocityGradientSolid**. We note that we consider no-slip extrapolated velocity for gradient computation. Finally, we evaluate accelerations due to various forces in **ContinuityEquation**, **ContinuityEquationSolid**, **PressureForces**, and **ComputeViscousForces**. We note that, in the evaluation of the continuity equation, we use extrapolated slip velocity on solids (Muta et al., 2020). We use the computed and extrapolated properties to integrate the particle properties and position. We use IPST to make the particles more uniform and update the properties using an accurate first-order correction. The IPST can be performed after every few timesteps. In our simulations, we perform shifting every 10 iterations.

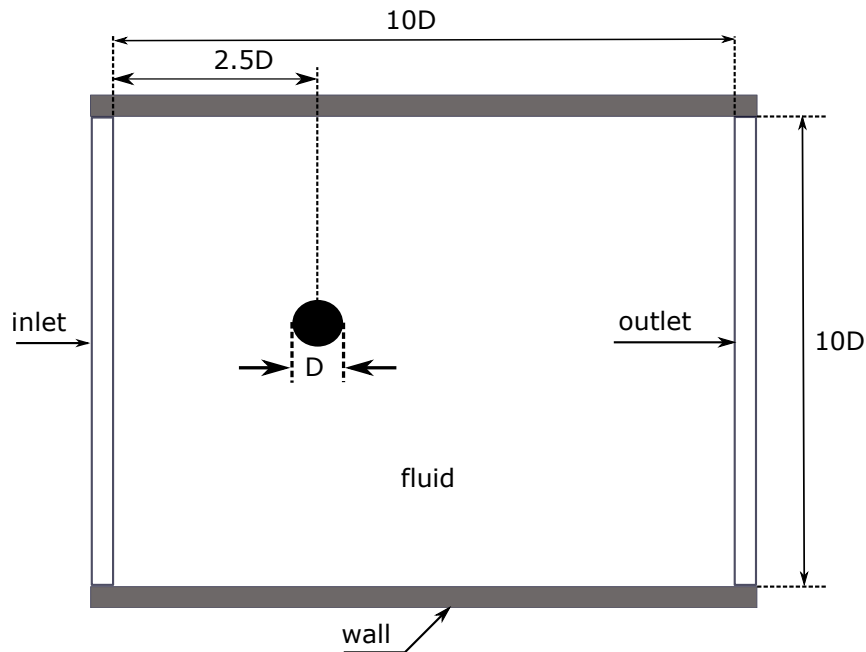


Figure 6.41 : Description of the domain of the flow past cylinder problem.

In order to show the accuracy achieved by the proposed algorithm, we solve the flow past a circular cylinder. We consider the domain shown in fig. 6.41. We consider inflow velocity $U = 1m/s$, Reynolds number $Re = 200$, and a cylinder of diameter $D = 2m$. We set the dynamic viscosity $\nu = UD/Re$. We discretize the domain with $\Delta x = D/40$. The

total particles in the domain are approximately $0.18M$. We simulate the problem using the algorithm 7 with the artificial speed of sound $c_o = 10m/s$ for $200sec$. We set the initial pressure $p_o = \rho_o c_o^2$, density $\rho_o = 1.0$, and velocity $\mathbf{u}_o = U\hat{i}$. We add an additional density damping proposed by Antuono et al. (2010) to the continuity equation with $\delta = 0.0625$, to reduce high-frequency pressure oscillations (see the variations of SOC schemes proposed in section 2.3.5).

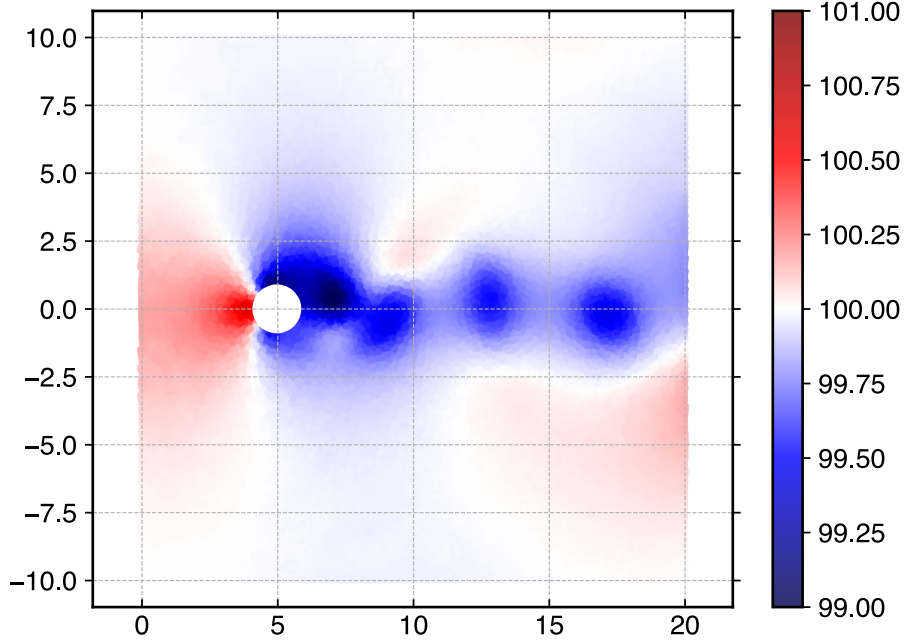


Figure 6.42 : Average pressure at $t = 100sec$.

In fig. 6.42 and fig. 6.43, we plot the average pressure and velocity magnitude at $t = 100sec$, respectively. We compute the average pressure $(p_{avg})_i = \sum p_j / N_{nbr}$, where the sum is taken over all the N_{nbr} neighbors of the i^{th} particle. Clearly, the solution is free from any high-frequency pressure oscillations. Furthermore, the pressure in the domain remains in the vicinity of the reference pressure $\rho_o c_o^2 = 100Pa$ for the entire simulation. In order to compute the coefficient of lift c_l and drag c_d for the cylinder, the force on the solid cylinder F_{solid} is determined by solving the momentum equation given by

$$\frac{F_{solid}}{m_{solid}} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}, \quad (6.15)$$

where m_{solid} is the mass of the solid particles. We use the SOC discretization for all the terms in the RHS of eq. (6.15) as discussed in section 2.3.3.

In fig. 6.44, we plot the variation of c_d and c_l with time for the present method with results of the EDAC scheme (Ramachandran et al., 2019) and the SISPH scheme (Muta et al., 2020). We obtain the mean c_d value of 1.65 and c_l value of 0.74, after the shedding is established. These values are closer to SISPH values, where a pressure Poisson equation

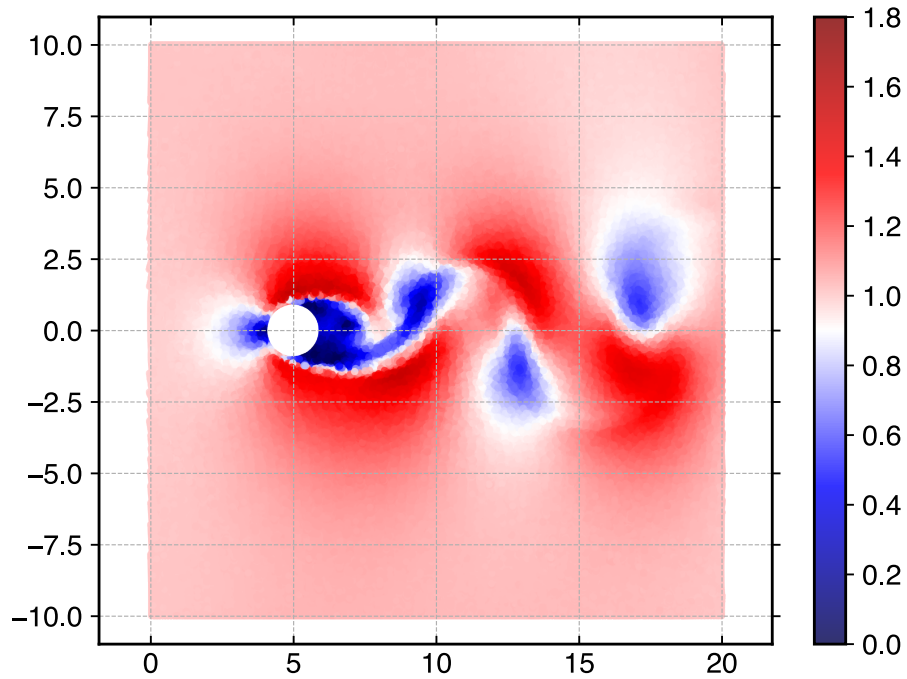


Figure 6.43 : Velocity magnitude at $t = 100\text{sec}$.

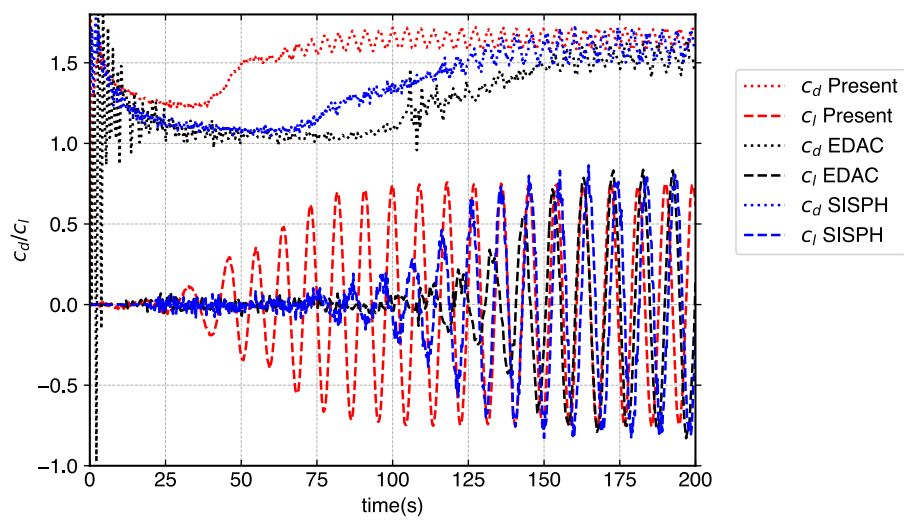


Figure 6.44 : c_d and c_l variation for the flow past a cylinder with time.

is solved to obtain pressure. Furthermore, both drag and lift coefficient values are free from disturbances. This example successfully demonstrates that the proposed algorithm works for a problem where all types of boundaries discussed in this work are present.

6.5 Summary

In this chapter, we use the MMS and construct MS for specific boundary conditions and domains. For a solid boundary, we verify methods for Neumann pressure, slip, and no-slip boundary conditions. In order to cover all the aspects of arbitrary geometries, we test the convergence on a straight, convex, and concave boundary. In the case of open boundaries, we consider a square domain with inlet and outlet regions simulating a wind tunnel. We manufacture solutions for inlets and outlets for both Neumann pressure and velocity. Additionally, we manufacture solutions depicting waves of pressure and velocity passing through inlet/outlet.

We show that the method proposed by Marrone et al. (2011) is convergent for all kinds of domain and boundary conditions on a solid boundary. Some other methods like Adami et al. (2012) and Colagrossi et al. (2003) are second-order convergent in inviscid flow and packed domains. Almost all boundary implementations are second-order on a straight boundary. In the case of open boundaries, the mirror and simple-mirror work well in the absence of a wave traveling through the boundary. The hybrid and do-nothing boundaries are bounded by the $O(M^2)$, where M is the Mach number of the flow. However, in the case of a wave traveling through the domain, the mirror and simple-mirror method diverges, and hybrid and do-nothing methods converge with second-order accuracy. Finally, we discuss an algorithm to apply these boundary conditions in order to get a convergent solver. We use the method proposed by Marrone et al. (2011) for solids and the hybrid method for inlet and outlet boundaries. We demonstrate the accuracy of the proposed algorithm by solving the flow past a circular cylinder. We achieve an accuracy close to the results obtained using incompressible SPH solvers.

Chapter 7

Conclusions and future work

In this work, we have addressed two of the grand-challenge problems in SPH (Vacondio et al., 2020). Firstly, we investigated the convergence of various existing weakly-compressible SPH schemes and proposed a second-order convergent scheme. Secondly, to the best of our knowledge, for the first time, we have identified convergent boundary condition implementations for SPH. We use the convergent boundary condition implementation and the scheme and propose a complete algorithm to simulate a flow past a circular cylinder problem. We list our contributions from this work as follows:

1. We systematically study various aspects that affect the convergence of a WCSPH scheme. We first compare various smoothing kernels that are widely used in SPH literature and select the best in terms of convergence and accuracy. We then use this kernel to compare different SPH formulations for gradient, divergence, and Laplacian approximation. We contrast the computational time, convergence, and conservation properties in all our comparisons.
2. We proposed several second-order convergent schemes. In order to achieve this, we propose four essential requirements.
 - (a) The density must be treated as a transport property.
 - (b) The integration volume must be evaluated using the particle density and mass of the particle.
 - (c) The operators must be discretized using SOC discretizations.
 - (d) PST must be used in order to maintain a uniform particle distribution, which ensures accurate approximations.
3. We compare the convergence of the proposed scheme and various existing WCSPH schemes by solving the Taylor-Green vortex problem. To the best of our knowl-

edge, we show the second-order convergence of a Lagrangian WCSPH scheme for the first time in SPH literature. We also show the conservation behavior of the proposed scheme and compare it with some of the widely used existing schemes. We observed that the proposed scheme is second-order convergent up to very high resolutions. However, the error is dominated by $O(M^2)$, where M is the Mach number of the flow, at high resolutions due to weakly-compressible assumption. We demonstrate the effect of the change of Mach number on the convergence and recommend using a high artificial speed of sound.

4. We propose variations of the SOC scheme. These variations are SOC versions of various existing weakly-compressible models. For example, transport velocity, EDAC, and Eulerian SPH. We show the convergence of these schemes by solving the Taylor-Green problem and also compare the conservation property of these variations.
5. We, to the best of our knowledge, for the first time, demonstrated the use of MMS to verify the convergence of the WCSPH scheme. We proposed a procedure to apply MMS to Lagrangian-based schemes. We suggested the initial particle distribution requirements and the minimum number of iterations required for a convergence study. We show that MMS is applicable to any shape of the domain and independent of the computational model used, for example, the use of transport velocity. This method takes far less time to produce the convergence plot compared to the method of exact solutions.
6. We show the application of MMS to obtain the convergence of various variations of the proposed scheme. We demonstrated the use of the MMS to obtain convergence at extreme resolutions, in 3D domains, and with fields of high-frequency variations. We also show that MMS can be used to find coding mistakes in an implementation or to identify terms causing the lack of convergence.
7. In order to apply solid boundary conditions accurately, one requires to capture the solid boundary features. We propose a hybrid technique to initialize the particles for both fluid and arbitrarily shaped solids simultaneously. Unlike existing methods, our method redistributes the particles in the domain such that the particles are smoothly aligned along the boundary of the solid object, maintaining a uniform particle density simultaneously. Our algorithm can be easily applied to any meshless method package.

8. As discussed before, the correct application of the boundary condition is a grand challenge. One of the reasons was the inability to verify the convergence of the boundary condition implementations. We, to the best of our knowledge, for the first time, verify the convergence of boundary condition implementation in SPH. We use the MMS to verify the convergence of Neumann pressure, slip, and no-slip, and inflow and outflow boundary conditions on straight, concave, and convex boundaries. In order to verify the boundary condition implementations, we manufacture solutions for different shaped domains which are applicable to a general Lagrangian-based codes.
9. We propose a complete second-order convergent scheme with solid and open boundaries that enables us to simulate realistic problems to obtain highly accurate solutions. We demonstrate the application of the algorithm by solving a flow past a circular cylinder problem. We compare the results with that of EDAC and SISPH and show that our results have much less noise.
10. All the results presented in this work are reproducible, and the source code is open-source.

It bears emphasis that all the work is implemented using an open-source tool PySPH (Ramachandran et al., 2021), and the results can be reproduced using automan (Ramachandran, 2018), an automation framework. Furthermore, all the work is freely available to reuse in various repositories at <https://gitlab.com/pypr>.

7.1 Future work

In this section, we list some directions for research in the future as follows:

1. We observe that the SPH discretization of operators involved in the momentum equation is either convergent or conservative. A convergent and conservative discretization of the momentum equation is one possible avenue for research.
2. The divergence of divergence-free fields is zero when the particles are arranged in a Cartesian grid. However, the error goes to a $O(10^{-4})$ for packed particle distribution. Methods to remove this error due to the particle distribution can be investigated in the future.
3. The application of kernel gradient correction in order to improve the convergence of the scheme affects the computation time by a factor of two. In order to speed

up the process, one can use an adaptive formulation, recently proposed by Muta et al. (2022). However, a second-order convergent adaptive SPH formulation is not known. In the future, we would like to use MMS to identify the root cause of the lack of convergence present in the current schemes and construct a second-order version of adaptive schemes. This would greatly motivate one to use the second-order accurate schemes in practical simulations.

4. Since the application of the kernel correction is essential for second-order convergent scheme, efficient ways to implement these corrections could be investigated in the future.
5. The SPH method is widely used to solve free-surface and interfacial flows. However, a second-order convergent implementation to apply these boundary conditions is not known. In the future, we would like to use MMS to identify and possibly construct a second-order convergent boundary implementation for free-surface and interfacial flows.
6. In the case of particle packing, we observed that in three dimensions, the particles placed on a rectangular grid do not have the lowest potential. However, this configuration is widely used to simulate flows in three dimensions. In the future, we would like to explore the effect of using a stable configuration like hexagonal packing to carry out simulations.

Appendix A

Supporting material

A.1 Skew-adjoint pressure gradient and velocity divergence operators

In this section, we show that the formulation for velocity divergence in eq. (1.20) and pressure gradient in eq. (1.24) are skew-adjoint. The inner product of the pressure gradient formulation and velocity is given by

$$\begin{aligned}\langle \langle \nabla p \rangle, \mathbf{u} \rangle &= \sum_j (p_j + p_i) \nabla W_{ij} \cdot \mathbf{u}_i \omega_j \\ &= - \sum_j (p_j + p_i) \nabla W_{ij} \cdot \mathbf{u}_j \omega_j \\ &= \frac{1}{2} \sum_j (p_j + p_i) \nabla W_{ij} \cdot (\mathbf{u}_i - \mathbf{u}_j) \omega_j,\end{aligned}\tag{A.1}$$

where the second step is performed by exchanging the dummy scripts and the third step is the average of the first two steps. Similarly, the inner product of the pressure and divergence of the velocity is given by

$$\begin{aligned}(p, \langle \nabla \cdot \mathbf{u} \rangle) &= - \sum_j p_i \nabla W_{ij} \cdot (\mathbf{u}_i - \mathbf{u}_j) \omega_j \\ &= - \sum_j p_j \nabla W_{ij} \cdot (\mathbf{u}_i - \mathbf{u}_j) \omega_j \\ &= - \frac{1}{2} \sum_j (p_i + p_j) \nabla W_{ij} \cdot (\mathbf{u}_i - \mathbf{u}_j) \omega_j.\end{aligned}\tag{A.2}$$

Therefore,

$$\langle \langle \nabla p \rangle, \mathbf{u} \rangle = - (p, \langle \nabla \cdot \mathbf{u} \rangle).\tag{A.3}$$

A.2 Instabilities inherent to the smoothing kernel

In this section, we discuss two well-known instabilities inherent to the smoothing kernels, viz. pairing and tensile instability. Swegle et al. (1995) observed that in the presence of tensile stress on a particle lattice, the particles rapidly clump together to create a hole. They propose that the unstable growth can be controlled if

$$W''T < 0, \quad (\text{A.4})$$

where W'' is the second derivative of the smoothing kernel and T is the total stress. Therefore, the time step must be set accordingly to achieve stability. Cubic spline kernels were found to have tensile instabilities when the smoothing length is kept equal to the particle spacing.

These tensile instabilities grow and particles pair up. This reduces the number of effective neighbors for a given particle and makes the simulation inaccurate. This issue is called pairing instability. Dehnen et al. (2012) proposed that the smoothing kernel with negative Fourier transform for some k wavenumber will inevitably trigger pairing instability. The Wendland class of kernels was found to be free from pairing instability. However, the approach of particle shifting either explicitly (in L-IPST-C scheme) or via transport velocity (in TV-C scheme) eliminates these instabilities for all types of smoothing kernels considered in this work.

A.3 Derivation of kernel gradient correction

In this section, we derive the correction matrices discussed in chapter 1. We use the discrete form of the Taylor-series expansion of a function f at \mathbf{x}_j about \mathbf{x}_i , given by

$$f_j = f_i - (\mathbf{x}_{ij} \cdot \nabla)f_i + \frac{1}{2}(\mathbf{x}_{ij} \cdot \nabla)^2 f_i + O(|\mathbf{x}_{ij}|^3), \quad (\text{A.5})$$

where $f_j = f(\mathbf{x}_j)$. In the following sections, we derive the different kernel gradient corrections.

A.3.1 Correction by Bonet et al. (1999)

In this section, we derive the kernel gradient correction proposed by Bonet et al. (1999). For a smoothing kernel to approximate the gradient with first-order consistency, it must satisfy

$$\sum_j (\mathbf{x}_j - \mathbf{x}_i) \otimes \nabla \tilde{W}_{ij} \omega_j = \mathbf{I},$$

where $\nabla\tilde{W}_{ij} = B_i\nabla W_{ij}$ is the corrected kernel gradient and B_i is the correction matrix. Therefore, we have ¹

$$\begin{aligned} \sum_j (\mathbf{x}_j - \mathbf{x}_i) \otimes B_i \nabla W_{ij} \omega_j &= \mathbf{I} \\ \implies \left[\sum_j (\mathbf{x}_j - \mathbf{x}_i) \otimes \nabla W_{ij} \omega_j \right] B_i^T &= \mathbf{I}. \end{aligned}$$

From the above equation, the correction matrix is evaluated as

$$B_i = \left[\sum_j \nabla W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \omega_j \right]^{-1}.$$

A.3.2 Correction by Liu et al. (2006)

Liu et al. (2006) use the Taylor-series expansion of a function in eq. (A.5). Taking the discrete convolution of the expansion with the kernel W_{ij} and its gradient ∇W_{ij} , we get

$$\sum_j f_j W_{ij} \omega_j = \sum_j f_i W_{ij} \omega_j - \sum_j (\mathbf{x}_{ij} \cdot \nabla) f_i W_{ij} \omega_j, \quad (\text{A.6})$$

ignoring the higher-order terms, and

$$\sum_j f_j \nabla W_{ij} \omega_j = \sum_j f_i \nabla W_{ij} \omega_j - \sum_j (\mathbf{x}_{ij} \cdot \nabla) f_i \nabla W_{ij} \omega_j. \quad (\text{A.7})$$

From the eq. (A.6) and eq. (A.7), we get

$$\begin{bmatrix} \sum_j f_j W_{ij} \omega_j \\ \sum_j f_j \nabla W_{ij} \omega_j \end{bmatrix} = \begin{bmatrix} \sum_j W_{ij} \omega_j & \sum_j \mathbf{x}_{ji} W_{ij} \omega_j \\ \sum_j \nabla W_{ij} \omega_j & \sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j \end{bmatrix} \begin{bmatrix} f_i \\ \nabla f_i \end{bmatrix}.$$

Considering, the correction kernel \tilde{W}_{ij} and kernel gradient $\nabla\tilde{W}_{ij}$, we get

$$\begin{bmatrix} \sum_j f_j W_{ij} \omega_j \\ \sum_j f_j \nabla W_{ij} \omega_j \end{bmatrix} = \begin{bmatrix} \sum_j W_{ij} \omega_j & \sum_j \mathbf{x}_{ji} W_{ij} \omega_j \\ \sum_j \nabla W_{ij} \omega_j & \sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j \end{bmatrix} \begin{bmatrix} \sum_j f_j \tilde{W}_{ij} \omega_j \\ \sum_j f_j \nabla \tilde{W}_{ij} \omega_j \end{bmatrix}.$$

Therefore, we have

$$\begin{bmatrix} W_{ij} \\ \nabla W_{ij} \end{bmatrix} = \begin{bmatrix} \sum_j W_{ij} \omega_j & \sum_j \mathbf{x}_{ji} W_{ij} \omega_j \\ \sum_j \nabla W_{ij} \omega_j & \sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j \end{bmatrix} \begin{bmatrix} \tilde{W}_{ij} \\ \nabla \tilde{W}_{ij} \end{bmatrix}.$$

Therefore, the correction matrix is given by

$$\begin{bmatrix} \tilde{W}_{ij} \\ \nabla \tilde{W}_{ij} \end{bmatrix} = L_i \begin{bmatrix} W_{ij} \\ \nabla W_{ij} \end{bmatrix} = \begin{bmatrix} \sum_j W_{ij} \omega_j & \sum_j \mathbf{x}_{ji} W_{ij} \omega_j \\ \sum_j \nabla W_{ij} \omega_j & \sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j \end{bmatrix}^{-1} \begin{bmatrix} W_{ij} \\ \nabla W_{ij} \end{bmatrix},$$

where L_i denotes the correction matrix. We note that L_i is $(n+1) \times (n+1)$ size matrix, where n is the dimension of the smoothing kernel.

¹See Gurtin (1982) for details about algebraic manipulations.

A.3.3 Correction by Huang et al. (2019)

Huang et al. (2019) proposed to evaluate the correction matrix without using the kernel gradient. The Taylor-series expansion of the function is convolved with the kernel as given in eq. (A.6) and the first moment of the smoothing kernel, given by

$$\sum_j f_j \mathbf{x}_{ij} W_{ij} \omega_j = \sum_j f_i \mathbf{x}_{ij} W_{ij} \omega_j - \sum_j (\mathbf{x}_{ij} \cdot \nabla) f_i \mathbf{x}_{ij} W_{ij} \omega_j. \quad (\text{A.8})$$

From eq. (A.6) and eq. (A.8), we get

$$\begin{bmatrix} \sum_j f_j W_{ij} \omega_j \\ \sum_j f_j \mathbf{x}_{ij} W_{ij} \omega_j \end{bmatrix} = \begin{bmatrix} \sum_j W_{ij} \omega_j & \sum_j \mathbf{x}_{ji} W_{ij} \omega_j \\ \sum_j \mathbf{x}_{ij} W_{ij} \omega_j & \sum_j \mathbf{x}_{ij} \otimes \mathbf{x}_{ji} \omega_j \end{bmatrix} \begin{bmatrix} f_i \\ \nabla f_i \end{bmatrix}.$$

Using similar manipulation as done in the previous section, we get

$$\begin{bmatrix} \tilde{W}_{ij} \\ \nabla \tilde{W}_{ij} \end{bmatrix} = H_i \begin{bmatrix} W_{ij} \\ \mathbf{x}_{ij} W_{ij} \end{bmatrix} = \begin{bmatrix} \sum_j W_{ij} \omega_j & \sum_j \mathbf{x}_{ji} W_{ij} \omega_j \\ \sum_j \mathbf{x}_{ij} W_{ij} \omega_j & \sum_j \mathbf{x}_{ij} \otimes \mathbf{x}_{ji} \omega_j \end{bmatrix}^{-1} \begin{bmatrix} W_{ij} \\ \mathbf{x}_{ij} W_{ij} \end{bmatrix},$$

where H_i is the correction matrix.

A.3.4 Correction by Fatehi et al. (2011)

In this section, we derive the correction proposed by Fatehi et al., 2011 for a second-order convergent Laplacian approximation. We also introduce the tensor notations for SPH that makes the comprehension of the formulations easier. We use the Taylor series expansion in eq. (A.5). We use tensor notation to represent vector \mathbf{x}_{ij} as x_{ij}^α , where i and j are the particle indices. We follow this notation since SPH approximation is performed using sum over all its neighbors j . Thus, we write the eq. (A.5) in this tensor notation as

$$f_j = f_i - x_{ij}^\alpha \partial^\alpha f_i + \frac{1}{2} x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma f_i + O(|\Delta x|^3). \quad (\text{A.9})$$

We note that the subscripts are SPH notations and the superscripts are tensor notation indices.

We write the Laplacian of the function f using the method proposed by Cleary et al. (1999) as

$$\langle \partial^n \partial^n f \rangle_i = \sum_j 2\omega_j (f_i - f_j) \frac{\partial^n W_{ij} x_{ij}^\eta}{r_{ij}^2}. \quad (\text{A.10})$$

The error E_i in the approximation is

$$E_i = \partial^n \partial^n f_i - \langle \partial^n \partial^n f \rangle_i. \quad (\text{A.11})$$

Using eq. (A.9) and eq. (A.10), we obtain the error

$$E_i = \partial^\theta \partial^\theta f_i - \sum_j 2\omega_j [x_{ij}^\alpha \partial^\alpha f_i - \frac{1}{2} x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma f_i + O(|\Delta x|^3)] \frac{\partial^n W_{ij} x_{ij}^\eta}{r_{ij}^2}. \quad (\text{A.12})$$

In the above equation, we can write $\partial^\theta \partial^\theta f_i = \delta^{\theta\mu} \partial^\theta \partial^\mu f_i$ and multiplying each term inside, we get

$$E_i = -\partial^\alpha f_i \sum_j 2\omega_j e_{ij}^\alpha e_{ij}^\eta \partial^n W_{ij} + \left(\mathcal{K}^{\beta\gamma} + \sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \frac{\partial^n W_{ij} x_{ij}^\eta}{r_{ij}^2} \right) \partial^\beta \partial^\gamma f_i + O(|\Delta x|^3), \quad (\text{A.13})$$

where \mathcal{K} is the Kronecker delta ². We can see that the first term is leading error term in the above equation. For a smoothing kernel W , the term $\sum_j \omega_j (\mathbf{x}_{ij} \otimes \mathbf{x}_{ij}) \nabla W_{ij}$ is the second moment of the kernel gradient. In a UP domain, the second moment is zero. However, the leading term of the error is the second moment scaled by $\frac{1}{|\mathbf{x}_{ij}|^2}$ which is still zero since it is a constant in a UP domain. Whereas, in the case of a PP domain, the leading term is non-zero and causes the approximation to deviate.

In the modified formulation proposed by Fatehi et al. (2011), the leading term is included in the approximation. The modified form is given by

$$\langle \partial^\theta \partial^\theta f_i \rangle_i = \sum_j 2\omega_j ((f_i - f_j) - x_{ij}^\alpha \langle \partial^\alpha u \rangle_i) \frac{\partial^n W_{ij} x_{ij}^\eta}{r_{ij}^2}. \quad (\text{A.14})$$

Using a similar algebraic manipulation, we write the error term as

$$E_i = \left(\sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \partial^\theta W_{ij} B_i^{T,\theta\alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\eta \partial^n W_{ij} + \mathcal{K}^{\beta\gamma} + \sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \frac{\partial^n W_{ij} x_{ij}^\eta}{r_{ij}^2} \right) \partial^\beta \partial^\gamma f_i + O(|\Delta x|^3), \quad (\text{A.15})$$

where $B_i^T = \left(\sum_j \nabla W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \omega_j \right)^{-T}$ is the correction matrix. Fatehi et al. (2011) also proposed a correction for the kernel gradient. Let us assume the correction $F_i^{\eta\mu}$ is applied to the kernel gradient. The modified equation is given by

$$\langle \partial^\theta \partial^\theta f_i \rangle_i = \sum_j 2\omega_j ((f_i - f_j) - x_{ij}^\alpha \langle \partial^\alpha f \rangle_i) \frac{F_i^{\eta\mu} \partial^\mu W_{ij} x_{ij}^\eta}{r_{ij}^2}. \quad (\text{A.16})$$

²We use a different symbol than the common used δ symbol as δ -SPH is a widely used scheme in SPH literature.

The error in the above equation is given by

$$E_i = \left(\sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \partial^\theta W_{ij} B_i^{T, \theta \alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\eta F_i^{\eta \mu} \partial^\mu W_{ij} + \mathcal{K}^{\beta \gamma} + \sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \frac{F_i^{\eta \mu} \partial^\mu W_{ij} x_{ij}^\eta}{r_{ij}^2} \right) \partial^\beta \partial^\gamma f_i + O(|\Delta x|^3). \quad (\text{A.17})$$

In order to make the approximation second-order accurate, we must have the coefficient of $\partial^\beta \partial^\gamma f_i$ equal to zero. Thus we get

$$\sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \partial^\theta W_{ij} B_i^{T, \theta \alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\eta F_i^{\eta \mu} \partial^\mu W_{ij} + \sum_j \omega_j e_{ij}^\beta e_{ij}^\gamma F_i^{\eta \mu} \partial^\mu W_{ij} x_{ij}^\eta = -\mathcal{K}^{\beta \gamma}. \quad (\text{A.18})$$

On inverting the system, we obtain

$$F_i^{\eta \mu} = - \left(\sum_j \omega_j \partial^\mu W_{ij} x_{ij}^\eta + \sum_j \omega_j r_{ij}^2 \partial^\theta W_{ij} B_i^{T, \theta \alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\eta \partial^\mu W_{ij} \right)^{-1}. \quad (\text{A.19})$$

The above equation is the correction matrix proposed by Fatehi et al. (2011) in a simple tensorial notation.

A.3.5 Correction by Korzilius et al. (2017)

In this section, we derive the correction proposed by Korzilius et al. (2017) for Laplacian formulation that involves the usage of the double derivative of the smoothing kernel. The Laplacian approximation with the double derivative of the smoothing kernel is given by

$$\langle \nabla^2 f \rangle_i = \sum_j (f_j - f_i) \nabla^2 W_{ij} \omega_j.$$

Using the Taylor-expansion in eq. (A.5), the error in the above approximation is given by

$$E_i = \nabla^2 f_i - \sum_j \left(-(\mathbf{x}_{ij} \cdot \nabla) f_i + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i + O(|\mathbf{x}_{ij}|^3) \right) \nabla^2 W_{ij} \omega_j.$$

In the tensor notation, we can write

$$\begin{aligned} E_i &= \partial^\gamma \partial^\gamma f_i + \sum_j x_{ij}^\beta \partial^\beta f_i \partial^\alpha \partial^\alpha W_{ij} \omega_j - \sum_j x_{ij}^\gamma x_{ij}^\epsilon \partial^\gamma \partial^\epsilon f_i \partial^\alpha \partial^\alpha W_{ij} \omega_j \\ &= \left[\sum_j x_{ij}^\beta \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\beta f_i - \left[\mathcal{K}^{\gamma \epsilon} - \sum_j x_{ij}^\gamma x_{ij}^\epsilon \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\gamma \partial^\epsilon f_i. \end{aligned}$$

Clearly, the coefficient of the gradient of the function is non-zero, resulting in a $O(h)$ error. Chen et al. (2000) included the first term in the above equation in the approximation and solved for all the derivatives of order 2 simultaneously. Consider the Taylor series expansion in eq. (A.5), convolving with all second-order derivative of the form $\partial^\alpha \partial^\beta W$ we get

$$\sum_j f_j \partial^x \partial^x W_{ij} \omega_j = \sum_j f_i \partial^x \partial^x W_{ij} \omega_j - \sum_j x_{ij}^\alpha \partial^\alpha f_i \partial^x \partial^x W_{ij} \omega_j + \sum_j \frac{1}{2} x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma f_i \partial^x \partial^x W_{ij} \omega_j,$$

$$\sum_j f_j \partial^x \partial^y W_{ij} \omega_j = \sum_j f_i \partial^x \partial^y W_{ij} \omega_j - \sum_j x_{ij}^\alpha \partial^\alpha f_i \partial^x \partial^y W_{ij} \omega_j + \sum_j \frac{1}{2} x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma f_i \partial^x \partial^y W_{ij} \omega_j,$$

and

$$\sum_j f_j \partial^y \partial^y W_{ij} \omega_j = \sum_j f_i \partial^y \partial^y W_{ij} \omega_j - \sum_j x_{ij}^\alpha \partial^\alpha f_i \partial^y \partial^y W_{ij} \omega_j + \sum_j \frac{1}{2} x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma f_i \partial^y \partial^y W_{ij} \omega_j.$$

Therefore, we have the linear system

$$\frac{1}{2} \begin{bmatrix} \sum_j x_{ij}^x x_{ij}^x \partial^x \partial^x W_{ij} \omega_j & \sum_j x_{ij}^x x_{ij}^y \partial^x \partial^x W_{ij} \omega_j & \sum_j x_{ij}^y x_{ij}^y \partial^x \partial^x W_{ij} \omega_j \\ \sum_j x_{ij}^x x_{ij}^x \partial^x \partial^y W_{ij} \omega_j & \sum_j x_{ij}^x x_{ij}^y \partial^x \partial^y W_{ij} \omega_j & \sum_j x_{ij}^y x_{ij}^y \partial^x \partial^y W_{ij} \omega_j \\ \sum_j x_{ij}^x x_{ij}^x \partial^y \partial^y W_{ij} \omega_j & \sum_j x_{ij}^x x_{ij}^y \partial^y \partial^y W_{ij} \omega_j & \sum_j x_{ij}^y x_{ij}^y \partial^y \partial^y W_{ij} \omega_j \end{bmatrix} \begin{bmatrix} \partial^x \partial^x f_i \\ \partial^x \partial^y f_i \\ \partial^y \partial^y f_i \end{bmatrix} \\ = \begin{bmatrix} \sum_j (f_j - f_i) \partial^x \partial^x W_{ij} \omega_j + \sum_j x_{ij}^\alpha \partial^\alpha f_i \partial^x \partial^x W_{ij} \omega_j \\ \sum_j (f_j - f_i) \partial^x \partial^y W_{ij} \omega_j + \sum_j x_{ij}^\alpha \partial^\alpha f_i \partial^x \partial^y W_{ij} \omega_j \\ \sum_j (f_j - f_i) \partial^y \partial^y W_{ij} \omega_j + \sum_j x_{ij}^\alpha \partial^\alpha f_i \partial^y \partial^y W_{ij} \omega_j \end{bmatrix},$$

where ∂^x, ∂^y denotes the x and y component of the partial derivatives. It must be noted that the gradient approximation on the RHS is not first-order consistent. The above equation can be written in a shorter form given by

$$\frac{1}{2} \sum_j \zeta_{ij}^\alpha \tilde{\Delta}^\beta W_{ij} \omega_j \tilde{\Delta}^\alpha f_i = \sum_j (f_j - f_i) \tilde{\Delta}^\beta W_{ij} \omega_j + \sum_j x_{ij}^\delta \partial^\delta f_i \tilde{\Delta}^\beta W_{ij} \omega_j,$$

where $\zeta_{ij} = [x_{ij}^x x_{ij}^x \quad x_{ij}^x x_{ij}^y \quad x_{ij}^y x_{ij}^y]^T$ and $\tilde{\Delta} = [\partial^x \partial^x \quad \partial^x \partial^y \quad \partial^y \partial^y]^T$. Korzilius et al. (2017) proposed to use the corrected gradient approximate to evaluate $\partial^\delta f_i$ in the above approximation resulting in a different correction matrix given by

$$K_i = \left[\frac{1}{2} \sum_j \zeta_{ij}^\alpha \tilde{\Delta}^\beta W_{ij} \omega_j - \sum_j \tilde{\Delta}^\beta x_{ji}^\alpha B_i^{\alpha\gamma} \omega_j \sum_j \frac{1}{2} \partial^\gamma W_{ij} \zeta_{ij}^\gamma \omega_j \right]^{-1}.$$

A.4 Derivation of error terms in SPH approximations

We use the Taylor-series expansion of an arbitrary field f at $\tilde{\mathbf{x}}$ about \mathbf{x} , given by

$$f(\tilde{\mathbf{x}}) = f(\mathbf{x}) - (\Delta \mathbf{x} \cdot \nabla) f(\mathbf{x}) + \frac{1}{2} (\Delta \mathbf{x} \otimes \Delta \mathbf{x}) : (\nabla \otimes \nabla) f(\mathbf{x}) + O(|\Delta \mathbf{x}|^3), \quad (\text{A.20})$$

where $\Delta \mathbf{x} = \mathbf{x} - \tilde{\mathbf{x}}$, $\nabla = \left[\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y} \quad \frac{\partial}{\partial z} \right]$. In the following subsection we present the derivation for various approximations.

A.4.1 Error in continuous SPH interpolation

The continuous SPH interpolation of an arbitrary field is given by

$$f(\mathbf{x}) \approx \int_{\Omega} f(\tilde{\mathbf{x}})W(\Delta\mathbf{x})d\tilde{\mathbf{x}}, \quad (\text{A.21})$$

where $W(\Delta\mathbf{x}) = W(\mathbf{x} - \tilde{\mathbf{x}}, h)$. Substituting the Taylor-series expansion from eq. (A.20), we get

$$f(\mathbf{x}) \approx \int_{\Omega} \left[f(\mathbf{x}) - (\Delta\mathbf{x} \cdot \nabla)f(\mathbf{x}) + \frac{1}{2}(\Delta\mathbf{x} \otimes \Delta\mathbf{x}) : (\nabla \otimes \nabla)f(\mathbf{x}) + O(|\Delta\mathbf{x}|^3) \right] W(\Delta\mathbf{x})d\tilde{\mathbf{x}}.$$

Expanding the above equation, we get

$$\begin{aligned} \langle f(\mathbf{x}) \rangle = & f(\mathbf{x}) \int_{\Omega} W(\Delta\mathbf{x})d\tilde{\mathbf{x}} - \nabla f(\mathbf{x}) \cdot \int_{\Omega} \Delta\mathbf{x}W(\Delta\mathbf{x})d\tilde{\mathbf{x}} + \\ & \frac{1}{2}(\nabla \otimes \nabla)f(\mathbf{x}) : \int_{\Omega} (\Delta\mathbf{x} \otimes \Delta\mathbf{x})W(\Delta\mathbf{x})d\tilde{\mathbf{x}} + \int_{\Omega} O(|\Delta\mathbf{x}|^3)W(\Delta\mathbf{x})d\tilde{\mathbf{x}}, \end{aligned} \quad (\text{A.22})$$

where $\langle \cdot \rangle$ denotes the approximation. The smoothing kernel is even function such that

$$W(-\Delta\mathbf{x}) = W(\Delta\mathbf{x}). \quad (\text{A.23})$$

Therefore,

$$\int_{\Omega} W(\Delta\mathbf{x})d\tilde{\mathbf{x}} = 1, \quad (\text{A.24})$$

and the odd moment of the kernel,

$$\int_{\Omega} \Delta\mathbf{x}W(\Delta\mathbf{x})d\tilde{\mathbf{x}} = 0. \quad (\text{A.25})$$

Substituting these in eq. (A.22), we get

$$\langle f(\mathbf{x}) \rangle = f(\mathbf{x}) + \underbrace{\frac{1}{2}(\nabla \otimes \nabla)f(\mathbf{x}) : \int_{\Omega} (\Delta\mathbf{x} \otimes \Delta\mathbf{x})W(\Delta\mathbf{x})d\tilde{\mathbf{x}}}_{O(h^2)} + \underbrace{\int_{\Omega} O(|\Delta\mathbf{x}|^3)W(\Delta\mathbf{x})d\tilde{\mathbf{x}}}_{O(h^3)},$$

where $\int_{\Omega} (\Delta\mathbf{x} \otimes \Delta\mathbf{x})W(\Delta\mathbf{x})d\tilde{\mathbf{x}}$ is the second moment of the smoothing kernel. The second term in the above equation is $O(h^2)$, and the last term is $O(h^3)$. Therefore, the continuous SPH interpolation is second-order accurate in space. Since the odd moments of the smoothing kernel are zero owing to the property in eq. (A.23), the integral in the last term of the above equation is also zero. Therefore, the next term is the next highest order of error i.e. $O(h^2)$.

A.4.2 Error in continuous SPH gradient interpolation

The continuous SPH interpolation of the gradient of an arbitrary function is given by

$$\langle \nabla f(\mathbf{x}) \rangle = \int_{\Omega} f(\mathbf{x}) \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}}. \quad (\text{A.26})$$

Substituting the Taylor-series expansion from eq. (A.20), we get

$$f(\mathbf{x}) \approx \int_{\Omega} \left[f(\mathbf{x}) - (\Delta \mathbf{x} \cdot \nabla) f(\mathbf{x}) + \frac{1}{2} (\Delta \mathbf{x} \otimes \Delta \mathbf{x}) : (\nabla \otimes \nabla) f(\mathbf{x}) + O(|\Delta \mathbf{x}|^3) \right] \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}}.$$

Expanding the above equation, we get

$$\begin{aligned} \langle \nabla f(\mathbf{x}) \rangle = & f(\mathbf{x}) \int_{\Omega} \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}} - \nabla f(\mathbf{x}) \cdot \int_{\Omega} \Delta \mathbf{x} \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}} + \\ & \frac{1}{2} (\nabla \otimes \nabla) f(\mathbf{x}) : \int_{\Omega} (\Delta \mathbf{x} \otimes \Delta \mathbf{x}) \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}} + \int_{\Omega} O(|\Delta \mathbf{x}|^3) \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}}. \end{aligned} \quad (\text{A.27})$$

The gradient of the smoothing kernel is odd function such that

$$\nabla W(-\Delta \mathbf{x}) = -\nabla W(\Delta \mathbf{x}). \quad (\text{A.28})$$

Therefore,

$$\int_{\Omega} \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}} = 0, \quad (\text{A.29})$$

and the odd moment of the gradient of the kernel,

$$\int_{\Omega} \Delta \mathbf{x} \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}} = -1. \quad (\text{A.30})$$

Substituting these in eq. (A.27), we get

$$\langle \nabla f(\mathbf{x}) \rangle = \nabla f(\mathbf{x}) + \underbrace{\frac{1}{2} (\nabla \otimes \nabla) f(\mathbf{x}) : \int_{\Omega} (\Delta \mathbf{x} \otimes \Delta \mathbf{x}) \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}}}_{O(h^2)} + \underbrace{\int_{\Omega} O(|\Delta \mathbf{x}|^3) \nabla W(\Delta \mathbf{x}) d\tilde{\mathbf{x}}}_{O(h^3)}. \quad (\text{A.31})$$

The second term in the above equation is $O(h^2)$, and the last term is $O(h^3)$. Therefore, the continuous SPH interpolation is second-order accurate in space.

A.4.3 Error estimation of discrete SPH gradient approximation

In this section, we derive error terms for different gradient approximations. We first formulation referred as GI is given by

$$\langle \nabla f_i \rangle_i = \sum_j f_j \nabla W_{ij} \omega_j, \quad (\text{A.32})$$

where $\nabla f_i = \nabla_i f(\mathbf{x}_i)$. Using the Taylor-series expansion for f_j about \mathbf{x}_i , we have

$$\begin{aligned} E_i &= \langle \nabla f \rangle_i - \nabla f_i \\ &= \sum_j f_j \nabla W_{ij} \omega_j - \nabla f_i \\ &= \sum_j \left[f_i - (\mathbf{x}_{ij} \cdot \nabla) f_i + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i + O(|\mathbf{x}_{ij}|^3) \right] \nabla W_{ij} \omega_j - \nabla f_i. \end{aligned}$$

Therefore, highest order of error is $O(0)$ due to $f_i \sum_j \nabla W_{ij} \omega_j$. Next, we consider the conservative formulation, referred as $G2$, given by

$$\langle \nabla f \rangle_i = \sum_j (f_j + f_i) \nabla W_{ij} \omega_j.$$

Using the Taylor-series expansion of f_j about \mathbf{x}_i , we get

$$E_i = \sum_j \left[2f_i - (\mathbf{x}_{ij} \cdot \nabla) f_i + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i + O(|\mathbf{x}_{ij}|^3) \right] \nabla W_{ij} \omega_j - \nabla f_i.$$

The error is again $O(0)$ as highest error doubled to $2f_i \sum_j \nabla W_{ij} \omega_j$. We now consider another conservative formulation, referred as $G3$, given by

$$\left\langle \frac{\nabla f}{\rho} \right\rangle_i = \sum_j m_j \left(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2} \right) \nabla W_{ij}.$$

The formulation $G2$ and $G3$ have conservative formulation. The error term of the $G3$ will be complex however an $O(0)$ error is expected. We consider the Liu correction matrix multiplied to $G2$, referred to as formulation $G4$, given by

$$\langle \nabla f \rangle_i = \sum_j (f_j + f_i) L_i \nabla W_{ij} \omega_j.$$

Therefore, the error is

$$\begin{aligned} E_i &= \sum_j \left[2f_i - (\mathbf{x}_{ij} \cdot \nabla) f_i + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i + O(|\mathbf{x}_{ij}|^3) \right] \nabla W_{ij} \omega_j - \nabla f_i \\ &= 2f_i \sum_j L_i \nabla W_{ij} \omega_j + \nabla f_i \sum_j (I - \mathbf{x}_{ji} \otimes L_i \nabla W_{ij} \omega_j) + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i L_i \nabla W_{ij} \omega_j. \end{aligned}$$

From the derivation of the Liu correction, we obtain that the first and the second term are zero. Therefore, the highest error is $\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i L_i \nabla W_{ij} \omega_j$. We next consider the zeroth-order convergent formulation referred to as $G5$, given by

$$\langle \nabla f \rangle_i = \sum_j (f_j - f_i) \nabla W_{ij} \omega_j.$$

Name	Formulation	Error	Order
<i>G1</i>	$\sum_j f_j \nabla W_{ij} \omega_j$	$f_i \sum_j \nabla W_{ij} \omega_j$	$O(0)$
<i>G2</i>	$\sum_j (f_j + f_i) \nabla W_{ij} \omega_j$	$2f_i \sum_j \nabla W_{ij} \omega_j$	$O(0)$
<i>G3</i>	$\sum_j m_j \left(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2} \right) \nabla W_{ij}$	$2f_i \sum_j \nabla W_{ij} \omega_j$	$O(0)$
<i>G4</i>	$\sum_j (f_j + f_i) L_i \nabla W_{ij} \omega_j$	$\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i L_i \nabla W_{ij} \omega_j$	$O(h^2)$
<i>G5</i>	$\sum_j (f_j - f_i) \nabla W_{ij} \omega_j$	$\left(\sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j - I \right) \nabla f_i$	$O(h)$
<i>G6</i>	$\sum_j (f_j - f_i) B_i \nabla W_{ij} \omega_j$	$\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i B_i \nabla W_{ij} \omega_j$	$O(h^2)$

Table A.1 : Different gradient formulations, their dominating error terms, and corresponding order for a uniform arrangement of particle.

Using the Taylor series expansion, the error is

$$\begin{aligned}
E_i &= \sum_j \left[-(\mathbf{x}_{ij} \cdot \nabla) f_i + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i + O(|\mathbf{x}_{ij}|^3) \right] \nabla W_{ij} \omega_j - \nabla f_i \\
&= \left(\sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j - I \right) \nabla f_i + \frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i \nabla W_{ij} \omega_j.
\end{aligned}$$

The highest error is $O(h)$. However, on applying the Bonet correction B_i to the kernel gradient, we get the formulation referred as *G6*, given by

$$\langle \nabla f_i \rangle_i = \sum_j (f_j - f_i) B_i \nabla W_{ij} \omega_j,$$

where the highest error goes to $\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i B_i \nabla W_{ij} \omega_j$. We summarize the error of all the gradient discretizations in the table A.1.

A.4.4 Error estimation of discrete SPH Laplacian approximation

The error terms in various formulations are derived along with the correction in appendix A.3.4 and appendix A.3.5. One more formulation using double summation is given by

$$\begin{aligned}
\langle \nabla^2 f \rangle_i &= \sum_j (\langle \nabla f \rangle_j - \langle \nabla f \rangle_i) \nabla W_{ij} \omega_j \\
&= \sum_j \left(\sum_k (f_k - f_j) B_j \nabla W_{jk} \omega_k - \sum_l (f_l - f_i) B_i \nabla W_{il} \omega_l \right) \nabla W_{ij} \omega_j,
\end{aligned}$$

where $\langle \nabla f \rangle_i$ are the first order consistent gradient approximation. The error in the above approximation is given by

$$E_i = \partial^n \partial^n f_i - \langle \partial^n \partial^n f \rangle_i.$$

Using expansion for gradient terms, we write

$$E_i = \partial^n \partial^n f_i - \sum_j \left(\sum_k (f_k - f_j) B_j^{\eta\alpha} \partial^\alpha W_{jk} \omega_k - \sum_l (f_l - f_i) B_i^{\eta\beta} \partial^\beta W_{il} \omega_l \right) \partial^n W_{ij} \omega_j.$$

Using the Taylor series expansion of f_k , f_j , and f_l about \mathbf{x}_i , we get

$$\begin{aligned} E_i &= \partial^n \partial^n f_i - \sum_j \left(\sum_k \left([f_i - x_{ik}^\alpha \partial^\alpha f_i + \frac{1}{2} x_{ik}^\beta x_{ik}^\gamma \partial^\beta \partial^\gamma f_i] - [f_i - x_{ij}^\alpha \partial^\alpha f_i + \frac{1}{2} x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma f_i] \right) \right. \\ &\quad \left. B_j^{\eta\lambda} \partial^\lambda W_{jk} \omega_k - \sum_l \left([f_i - x_{il}^\alpha \partial^\alpha f_i + \frac{1}{2} x_{il}^\beta x_{il}^\gamma \partial^\beta \partial^\gamma f_i] - f_i \right) B_i^{\eta\lambda} \partial^\lambda W_{il} \omega_l \right) \partial^n W_{ij} \omega_j \\ &= \partial^n \partial^n f_i - \sum_j \left(\sum_k \left([(x_{ij}^\alpha - x_{ik}^\alpha) \partial^\alpha f_i + \frac{1}{2} (x_{ik}^\beta x_{ik}^\gamma - x_{ij}^\beta x_{ij}^\gamma) \partial^\beta \partial^\gamma f_i] \right) \right. \\ &\quad \left. B_j^{\eta\lambda} \partial^\lambda W_{jk} \omega_k - \sum_l \left([-x_{il}^\alpha \partial^\alpha f_i + \frac{1}{2} x_{il}^\beta x_{il}^\gamma \partial^\beta \partial^\gamma f_i] B_i^{\eta\lambda} \partial^\lambda W_{il} \omega_l \right) \right) \partial^n W_{ij} \omega_j \\ &= \sum_j \left(- \sum_k x_{jk}^\alpha B_j^{\eta\lambda} \partial^\lambda W_{jk} \omega_k \partial^n W_{ij} \omega_j + \sum_l x_{il}^\alpha B_i^{\eta\lambda} \partial^\lambda W_{il} \omega_l \partial^n W_{ij} \omega_j \right) \partial^\alpha f_i + \\ &\quad \frac{1}{2} \left(\mathcal{K}^{\beta\gamma} + \sum_k -(x_{ik}^\beta x_{ik}^\gamma - x_{ij}^\beta x_{ij}^\gamma) B_j^{\eta\lambda} \partial^\lambda W_{jk} \omega_k \partial^n W_{ij} \omega_j - \sum_l x_{il}^\beta x_{il}^\gamma B_i^{\eta\lambda} \partial^\lambda W_{il} \omega_l \partial^n W_{ij} \omega_j \right) \partial^\beta \partial^\gamma f_i. \end{aligned}$$

Therefore, the first term in the above equation dominates making the formulation $O(h)$ accurate. However, on applying the correction to the kernel gradient, given by

$$\begin{aligned} \langle \nabla^2 f \rangle_i &= \sum_j (\langle \nabla f \rangle_j - \langle \nabla f \rangle_i) \nabla W_{ij} \omega_j \\ &= \sum_j \left(\sum_k (f_k - f_j) B_j \nabla W_{jk} \omega_k - \sum_l (f_l - f_i) B_i \nabla W_{il} \omega_l \right) B_i \nabla W_{ij} \omega_j. \end{aligned}$$

The first term equals in the error expression equals to zero making the formulation $O(h^2)$ accurate. In table A.2 we tabulate different Laplacian formulation their dominating error term and order of convergence.

Name	Formulation	Error	Order
L1	$\sum_j 2 \frac{(f_i - f_j)}{r_{ij}} \mathbf{e}_{ij} \cdot \nabla W_{ij} \omega_j$	$-\partial^\alpha f_i \sum_j 2 \omega_j e_i^\alpha e_{ij}^\eta \partial^\eta W_{ij}$	$O(0)$
L2	$\sum_j 2 \omega_j \left(\frac{(f_i - f_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla f \rangle_i \right) \mathbf{e}_{ij} \cdot \nabla W_{ij}$	$\left(\sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma W_{ij} \mathbf{B}_i^{T, \theta \alpha} \sum_j \omega_j e_i^\alpha e_{ij}^\eta \partial^\eta W_{ij} + \mathcal{I} \mathcal{C}^{\beta \gamma} + \sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \frac{\partial^\eta W_{ij} x_{ij}^\eta}{r_{ij}^2} \right) \partial^\beta \partial^\gamma f_i$	$O(h)$
L3	$\sum_j 2 \omega_j \left(\frac{(f_i - f_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla f \rangle_i \right) \mathbf{e}_{ij} \cdot F_i \nabla W_{ij}$	$O(\Delta x ^3) \frac{\partial^\eta W_{ij} x_{ij}^\eta}{r_{ij}^2}$	$O(h^2)$
L4	$\sum_j (f_j - f_i) \nabla^2 W_{ij} \omega_j$	$\left[\sum_j x_{ij}^\beta \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\beta f_i$	$O(0)$
L5	$\sum_j [(f_j - f_i) - \mathbf{x}_{ji} \cdot \langle \nabla f \rangle_i] \nabla^2 W_{ij} \omega_j$	$\left[\sum_j x_{ij}^\gamma \partial^\alpha \partial^\alpha W_{ij} \omega_j \mathbf{B}_i^{T, \epsilon \alpha} \sum_j \omega_j e_i^\alpha e_{ij}^\eta \partial^\eta W_{ij} + \mathcal{I} \mathcal{C}^{\gamma \epsilon} - \sum_j x_{ij}^\gamma x_{ij}^\epsilon \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\gamma \partial^\epsilon f_i$	$O(h)$
L6	$\sum_j [(f_j - f_i) - \mathbf{x}_{ji} \cdot \langle \nabla f \rangle_i] K_i \nabla^2 W_{ij} \omega_j$	$O(\Delta x ^3) \partial^\alpha \partial^\alpha W_{ij} \omega_j$	$O(h^2)$
L7	$\sum_j (\langle \nabla f \rangle_j - \langle \nabla f \rangle_i) \nabla W_{ij} \omega_j$	$O(\Delta x ^3) \mathbf{B}_i^{\eta \lambda} \mathbf{B}_j^{\gamma \lambda} \partial^\lambda W_{ij} \omega_j \partial^\eta W_{ij} \omega_j$	$O(h^2)$

Table A.2 : Different Laplacian formulations, their dominating error terms, and corresponding order for uniform arrangement of particles.

A.4.5 Quadrature error in discrete SPH gradient approximation

In this section, we show the error estimation due to discrete approximation of the integration in the continuous SPH interpolation done by Quinlan et al. (2006) in detail. For a general function f defined in $[x_1, x_n]$ in 1D, using the second Euler-Maclaurin formula, we can write

$$\Delta x \sum_{j=1}^n f_j = \int_{x_1-\Delta x/2}^{x_n+\Delta x/2} f(x)dx + \sum_{k=1}^{\infty} \frac{B_{2k}\Delta x^{2k}}{(2k)!} (1 - 2^{-2k+1}) (f_{(n+1/2)}^{(2k-1)} - f_{1/2}^{(2k-1)}), \quad (\text{A.33})$$

where $f_j = f(x_1 + j\Delta x)$, B_{2k} is the Bernoulli numbers, and f^k is the k^{th} derivative of the function f evaluated at the edge of the compact support smoothing kernel. Consider the function $f(x) = A(x)W'(x_a - x) = A(x)W'$, where $x = x_a$ is the position of the kernel. The discrete form is given by $f_j = A_j W'(x_a - x_j) = A_j W'_j$. Substituting this in the above equation, we get

$$\begin{aligned} \sum_{j=1}^n A_j W'_j \Delta x &= \int_{x_1-\Delta x/2}^{x_n+\Delta x/2} A(x)W' dx + \\ &\sum_{k=1}^{\infty} \frac{B_{2k}\Delta x^{2k}}{(2k)!} (1 - 2^{-2k+1}) ((AW')_{(n+1/2)}^{(2k-1)} - (AW')_{1/2}^{(2k-1)}). \end{aligned}$$

The first term on the LHS is the discrete SPH approximation of the function in 1D, and the first term on the RHS is the continuous SPH interpolation. Therefore, the last term is the exact quadrature error. Furthermore, the smoothing kernel has some of the non-zero lower-order derivatives at kernel boundary; therefore $(AW')_{(n+1/2)}^{(2k-1)} - (AW')_{1/2}^{(2k-1)}$ is zero for $2k + 1 \leq \beta - 1$. In this derivation, cubic spline kernel is considered for which $\beta = 2$. Considering the first non-zero term with $2k = \beta + 2$, and the position corresponding to the boundary of the kernel centered at x_a i.e. $\{x_a + 2h, x_a - 2h\}$, we get

$$\begin{aligned} E_i &= \sum_{j=1}^n A_j W'_j \Delta x - \int_{x_1-\Delta x/2}^{x_n+\Delta x/2} A(x)W' dx \\ &= \Delta x^{\beta+2} \frac{B_{\beta+2}}{(\beta+2)!} (1 - 2^{-\beta-1}) [(AW')_{x=x_a+2h}^{(\beta+1)} - (AW')_{x=x_a-2h}^{(\beta+1)}] + O(\Delta x^{\beta+4}). \end{aligned} \quad (\text{A.34})$$

Consider the Taylor-series expansion of the function $A(x)$ about x_a , given by

$$A(x) = A_a + A'_a \Delta x + A''_a \Delta x^2 / 2 + O(\Delta x^3).$$

Therefore,

$$\begin{aligned} (AW')^{(\beta+1)} &= ((A_a + A'_a \Delta x + A''_a \Delta x^2 / 2 + O(\Delta x^3))W')^{(\beta+1)} \\ &= (A_a W')^{(\beta+1)} + (A'_a \Delta x W')^{(\beta+1)} + (A''_a \Delta x^2 W' / 2)^{(\beta+1)} + (O(\Delta x^3)W')^{(\beta+1)}. \end{aligned}$$

Name	Formulation	Error	Order
$G1$	$\sum_j f_j \nabla W_{ij} \omega_j$	$f_i \sum_j \nabla W_{ij} \omega_j$	$ \nabla f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$G2$	$\sum_j (f_j + f_i) \nabla W_{ij} \omega_j$	$2f_i \sum_j \nabla W_{ij} \omega_j$	$ \nabla f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$G3$	$\sum_j m_j \left(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2}\right) \nabla W_{ij}$	$2f_i \sum_j \nabla W_{ij} \omega_j$	$ \nabla f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$G4$	$\sum_j (f_j + f_i) L_i \nabla W_{ij} \omega_j$	$\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i L_i \nabla W_{ij} \omega_j$	$ \nabla^3 f_i O\left(h^2 \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$G5$	$\sum_j (f_j - f_i) \nabla W_{ij} \omega_j$	$\left(\sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j - I\right) \nabla f_i$	$ \nabla f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$G6$	$\sum_j (f_j - f_i) B_i \nabla W_{ij} \omega_j$	$\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i B_i \nabla W_{ij} \omega_j$	$ \nabla^3 f_i O\left(h^2 \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$

Table A.3 : Different gradient formulations, their dominating quadrature error terms, and corresponding order for uniform arrangement of particles.

The derivative of the 1D smoothing kernel is given by, $W' = h^{-2} \hat{W}'$. Therefore, the above equation is written as

$$(AW')^{(\beta+1)} = \frac{A_a}{h^{(\beta+3)}} \hat{W}^{(\beta+2)} + \frac{A'_a}{h^{(\beta+2)}} \left[q \hat{W}^{(\beta+2)} + (\beta+1) \hat{W}^{(\beta+1)} \right] \\ + \frac{A''_a}{2h^{(\beta+1)}} \left[q^2 \hat{W}^{(\beta+2)} + 2q(\beta+1) \hat{W}^{(\beta+1)} + (\beta+1)\beta \hat{W}^{(\beta)} \right] + O\left(\frac{1}{h^{(\beta)}}\right),$$

where $q = \Delta x/h$. Using the above expression, we obtain

$$\left[(AW')_{x=x_a+2h}^{(\beta+1)} - (AW')_{x=x_a-2h}^{(\beta+1)} \right] \\ = \frac{A'_a}{h^{\beta+2}} \left[4 \hat{W}_{q=2}^{(\beta+2)} + 2(\beta+1) \hat{W}_{q=2}^{(\beta+1)} \right] \\ + \frac{A''_a}{3h^\beta} \left[8 \hat{W}_{q=2}^{(\beta+2)} + 12(\beta+1) \hat{W}_{q=2}^{(\beta+1)} + 6(\beta+1)\beta \hat{W}_{q=2}^{(\beta)} + (\beta+1)\beta(\beta-1) \hat{W}_{q=2}^{(\beta-1)} \right] \\ + O\left(\frac{1}{h^{\beta-2}}\right).$$

Putting this term in the eq. (A.34), we get

$$E_i = \left(\frac{\Delta x}{h}\right)^{\beta+2} \frac{B_{\beta+2}}{(\beta+2)!} (1 - 2^{-\beta-1}) \\ \times \left\{ A'_a \left[4 \hat{W}_{q=2}^{(\beta+2)} + 2(\beta+1) \hat{W}_{q=2}^{(\beta+1)} \right] + O(h^2) \right\} + O\left(\left[\frac{\Delta x}{h}\right]^{\beta+4}\right).$$

Therefore, in addition to the $O(h^2)$ error, the quadrature introduces $O\left(\left(\frac{\Delta x}{h}\right)^{\beta+2}\right)$ error. In the case of the smoothing kernel having odd β the error becomes $O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$. In a similar manner, we can derive the quadrature error for various other terms as done by Fatehi et al. (2011). In the table A.3 and table A.4, we tabulate the quadrature error from different gradient and Laplacian formulations.

Name	Formulation	Error	Order
L1	$\sum_j 2 \frac{(f_i - f_j)}{r_{ij}} \mathbf{e}_{ij} \cdot \nabla W_{ij} \omega_j$	$-\partial^\alpha f_i \sum_j 2 \omega_j e_{ij}^\alpha e_{ij}^\eta \partial^\eta W_{ij}$	$ \nabla^2 f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
L2	$\sum_j 2 \omega_j \left(\frac{(f_i - f_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla f \rangle_i \right) \mathbf{e}_{ij} \cdot \nabla W_{ij}$	$\left(\sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma W_{ij} \mathbf{B}_i^{T, \beta\alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\eta \partial^\eta W_{ij} + \mathcal{K}^{\beta\gamma} + \sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \frac{\partial^\eta W_{ij} x_{ij}^\eta}{r_{ij}^2} \right) \partial^\beta \partial^\gamma f_i$	$ \nabla^3 f_i O\left(h^2 \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
L3	$\sum_j 2 \omega_j \left(\frac{(f_i - f_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla f \rangle_i \right) \mathbf{e}_{ij} \cdot F_i \nabla W_{ij}$	$O(\Delta x^3) \frac{\partial^\eta W_{ij} x_{ij}^\eta}{r_{ij}^2}$	$ \nabla^3 f_i O\left(h^2 \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
L4	$\sum_j (f_j - f_i) \nabla^2 W_{ij} \omega_j$	$\left[\sum_j x_{ij}^\beta \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\beta f_i$	$ \nabla^2 f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
L5	$\sum_j \left[(f_j - f_i) - \mathbf{x}_{ji} \cdot \langle \nabla f \rangle_i \right] \nabla^2 W_{ij} \omega_j$	$\left[\sum_j x_{ij}^\gamma \partial^\alpha \partial^\alpha W_{ij} \omega_j \mathbf{B}_i^{T, \epsilon\alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\eta \partial^\eta W_{ij} + \mathcal{K}^{\gamma\epsilon} - \sum_j x_{ij}^\gamma x_{ij}^\epsilon \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\gamma \partial^\epsilon f_i$	$ \nabla^3 f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
L6	$\sum_j \left[(f_j - f_i) - \mathbf{x}_{ji} \cdot \langle \nabla f \rangle_i \right] K_i \nabla^2 W_{ij} \omega_j$	$O(\Delta x^3) \partial^\alpha \partial^\alpha W_{ij} \omega_j$	$ \nabla^3 f_i O\left(\left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
L7	$\sum_j \langle \nabla f \rangle_j - \langle \nabla f \rangle_i \nabla W_{ij} \omega_j$	$O(\Delta x^3) \mathbf{B}_i^{\eta\lambda} \partial^\lambda W_{ij} \omega_j \partial^\eta W_{ij} \omega_j$	$ \nabla^3 f_i O\left(h^2 \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$

Table A.4 : Different Laplacian formulations, their dominating quadrature error terms, and corresponding order for uniform arrangement of particles.

A.4.6 Quadrature error in discrete SPH gradient approximation on a non-uniform particle distribution

In this section, we derive an error in gradient approximation for an irregular arrangement of particle done by Fatehi et al. (2011). Consider the perturbation \mathbf{d}_i on the particles placed on Cartesian grid with spacing Δs . The perturbation \mathbf{d}_i introduced using a Normal distribution given by

$$\tilde{\mathbf{d}}_i = 0.25\Delta s\mathbf{z},$$

where $\mathbf{z} \in \{[-0.5, 0.5], [-0.5, 0.5], [0.5, 0.5]\}$ is the random variable generated from a normal distribution with zero means. The new position is given by

$$\mathbf{x}_{ij} = \mathbf{x}_{ij}^{reg} + \tilde{\mathbf{d}}_{ij},$$

where $\tilde{\mathbf{d}}_{ij} = \tilde{\mathbf{d}}_i - \tilde{\mathbf{d}}_j$. The kernel function using the Taylor-series approximation is given by

$$\begin{aligned} W(\mathbf{x}_{ij}) &= W(\mathbf{x}_{ij}^{reg} + \tilde{\mathbf{d}}_{ij}) \\ &= W(\mathbf{x}_{ij}^{reg}) + \tilde{\mathbf{d}}_{ij} \cdot \nabla W(\mathbf{x}_{ij}^{reg}) + \text{HOT}, \end{aligned}$$

where HOT are the higher-order term. Similarly, the gradient of the kernel is approximated as

$$\nabla W(\mathbf{x}_{ij}) = \nabla W(\mathbf{x}_{ij}^{reg}) + \tilde{\mathbf{d}}_{ij} \cdot \nabla \nabla W(\mathbf{x}_{ij}^{reg}) + \text{HOT}.$$

Therefore, the terms that are zero in the quadrature error due to symmetry of the kernel in the uniform arrangement of particles are non-zero, thus produces error. The quadrature error for

$$\sum_j \nabla W_{ij} \omega_j \approx |\tilde{\mathbf{d}}_i| \sum_j \nabla \cdot \nabla W_{ij} \omega_j \approx O\left(\frac{|\tilde{\mathbf{d}}_i|}{h^2} \left(\frac{\Delta s}{h}\right)^{\beta-1}\right).$$

In the appendix A.4.6, we list the errors for various summations derived by Fatehi et al. (2011). In table A.6 and table A.7, we tabulate the order of error for various formulations.

Summarize	Order
$\sum_j \nabla W_{ij} \omega_j$	$O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
B_i	$O(1)$
$I - B_i^{-1}$	$O\left(\left(\frac{\Delta s}{h}\right)^{\beta+1}\right) + O\left(\frac{ \tilde{\mathbf{d}}_i }{h} \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$\sum_j \mathbf{x}_{ij} \mathbf{x}_{ij} \nabla W_{ij} \omega_j$	$O\left(\tilde{\mathbf{d}}_i \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$\sum_j \mathbf{x}_{ij} \mathbf{x}_{ij} \mathbf{x}_{ij} \nabla W_{ij} \omega_j$	$O(h^2) + O\left(\tilde{\mathbf{d}}_i \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$\sum_j \nabla \nabla W_{ij} \omega_j$	$O\left(\frac{1}{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right) + O\left(\frac{ \tilde{\mathbf{d}}_i }{h^3} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
$\sum_j \mathbf{x}_{ij} \nabla \nabla W_{ij} \omega_j$	$O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$

Table A.5 : Order of quadrature errors for different SPH summations for an irregular arrangement of particles.

Name	Formulation	Error	Order
$G1$	$\sum_j f_j \nabla W_{ij} \omega_j$	$f_i \sum_j \nabla W_{ij} \omega_j$	$ f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
$G2$	$\sum_j (f_j + f_i) \nabla W_{ij} \omega_j$	$2f_i \sum_j \nabla W_{ij} \omega_j$	$ f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
$G3$	$\sum_j m_j \left(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2}\right) \nabla W_{ij}$	$2f_i \sum_j \nabla W_{ij} \omega_j$	$ f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
$G4$	$\sum_j (f_j + f_i) L_i \nabla W_{ij} \omega_j$	$\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i L_i \nabla W_{ij} \omega_j$	$ \nabla^2 f_i O\left(\tilde{\mathbf{d}}_i \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
$G5$	$\sum_j (f_j - f_i) \nabla W_{ij} \omega_j$	$(\sum_j \nabla W_{ij} \otimes \mathbf{x}_{ji} \omega_j - I) \nabla f_i$	$ \nabla f_i \left(O\left(\left(\frac{\Delta s}{h}\right)^{\beta+1}\right) + O\left(\frac{ \tilde{\mathbf{d}}_i }{h} \left(\frac{\Delta x}{h}\right)^{\beta+1}\right) \right)$
$G6$	$\sum_j (f_j - f_i) B_i \nabla W_{ij} \omega_j$	$\frac{1}{2} (\mathbf{x}_{ij} \cdot \nabla)^2 f_i B_i \nabla W_{ij} \omega_j$	$ \nabla^2 f_i O\left(\tilde{\mathbf{d}}_i \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$

Table A.6 : Different gradient formulations, their dominating quadrature error terms, and corresponding order for non-uniform particle arrangement.

Name	Formulation	Error	Order
L1	$\sum_j 2 \frac{(f_i - f_j)}{r_{ij}} \mathbf{e}_{ij} \cdot \nabla W_{ij} \omega_j$	$-\partial^\alpha f_i \sum_j 2 \omega_j e_{ij}^\alpha e_{ij}^\beta \partial^\beta W_{ij}$	$ \nabla f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
L2	$\sum_j 2 \omega_j \left(\frac{(f_i - f_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla f \rangle_i \right) \mathbf{e}_{ij} \cdot \nabla W_{ij}$	$\left(\sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \partial^\beta \partial^\gamma W_{ij} \mathbf{B}_i^{T, \beta\alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\beta \partial^\beta W_{ij} + \mathcal{K}^{\beta\gamma} + \sum_j \omega_j x_{ij}^\beta x_{ij}^\gamma \frac{\partial^\beta W_{ij} x_{ij}^\beta}{r_{ij}^2} \right) \partial^\beta \partial^\gamma f_i$	$ \nabla^2 f_i \left(O\left(\left(\frac{\Delta x}{h}\right)^{\beta+1}\right) + O\left(\frac{ \tilde{\mathbf{d}}_i }{h} \left(\frac{\Delta x}{h}\right)^{\beta+1}\right) \right)$
L3	$\sum_j 2 \omega_j \left(\frac{(f_i - f_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot \langle \nabla f \rangle_i \right) \mathbf{e}_{ij} \cdot \mathbf{F}_i \cdot \nabla W_{ij}$	$O(\Delta x ^3) \frac{\partial^\beta W_{ij} x_{ij}^\beta}{r_{ij}^2}$	$ \nabla^3 f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h} \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
L4	$\sum_j (f_j - f_i) \nabla^2 W_{ij} \omega_j$	$\left[\sum_j x_{ij}^\beta \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\beta f_i$	$ \nabla^2 f_i \left(O\left(\frac{1}{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right) + O\left(\frac{ \tilde{\mathbf{d}}_i }{h^3} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right) \right)$
L5	$\sum_j \left[(f_j - f_i) - \mathbf{x}_{ji} \cdot \langle \nabla f \rangle_i \right] \nabla^2 W_{ij} \omega_j$	$\left[\sum_j x_{ij}^\beta \partial^\alpha \partial^\alpha W_{ij} \omega_j \mathbf{B}_i^{T, \beta\alpha} \sum_j \omega_j e_{ij}^\alpha e_{ij}^\beta \partial^\beta W_{ij} + \mathcal{K}^{\beta\gamma} - \sum_j x_{ij}^\beta x_{ij}^\gamma \partial^\alpha \partial^\alpha W_{ij} \omega_j \right] \partial^\beta \partial^\gamma f_i$	$ \nabla f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h^2} \left(\frac{\Delta x}{h}\right)^{\beta-1}\right)$
L6	$\sum_j \left[(f_j - f_i) - \mathbf{x}_{ji} \cdot \langle \nabla f \rangle_i \right] K_i \nabla^2 W_{ij} \omega_j$	$O(\Delta x ^3) \partial^\alpha \partial^\alpha W_{ij} \omega_j$	$ \nabla^3 f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h} \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$
L7	$\sum_j \langle \nabla f \rangle_j - \langle \nabla f \rangle_i \nabla W_{ij} \omega_j$	$O(\Delta x ^3) \mathbf{B}_i^{T, \beta\alpha} \mathbf{B}_i^{T, \beta\alpha} \partial^\beta W_{ij} \omega_j \partial^\beta W_{ij} \omega_j$	$ \nabla^3 f_i O\left(\frac{ \tilde{\mathbf{d}}_i }{h} \left(\frac{\Delta x}{h}\right)^{\beta+1}\right)$

Table A.7 : Different Laplacian formulations, their dominating quadrature error terms, and corresponding order for non-uniform particle arrangement.

A.5 Weakly-compressible SPH schemes

In the weakly-compressible SPH (WCSPH) schemes, eq. (1.40) is discretized using the formulations discussed in section 1.3. Furthermore, two widely used artificial EOS are

$$p = p_o \left(\left(\frac{\rho}{\rho_o} \right)^7 - 1 \right), \quad (\text{A.35})$$

where p_o and ρ_o are reference pressure and density, and

$$p = c_o^2 (\rho - \rho_o). \quad (\text{A.36})$$

In SPH, the weakly-compressible assumption is used in the following schemes.

A.5.1 Standard scheme

In the first application of a weakly-compressible model using SPH to solve fluid flows proposed by Morris et al. (1997), the continuity equation in eq. (1.40a) is discretized as

$$\frac{d\rho_i}{dt} = \sum_j (\mathbf{u}_i - \mathbf{u}_j) \cdot \nabla W_{ij} \omega_j, \quad (\text{A.37})$$

and the momentum equation is discretized as

$$\frac{d\mathbf{u}_i}{dt} = \sum_j -m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} + 2m_j \frac{(\mu_i + \mu_j)(\mathbf{u}_i - \mathbf{u}_j)}{\rho_i \rho_j r_{ij}} \mathbf{e}_{ij} \cdot \nabla W_{ij}, \quad (\text{A.38})$$

where μ is the dynamic viscosity of the fluid. The discretization conserves the linear and angular momentum for particles having constant mass. The pressure is linked to the density using eq. (A.35), where $p_o = \rho_o c_o^2$.

A.5.2 δ -SPH scheme

The standard scheme is prone to high-frequency pressure oscillations due to the high sensitivity of pressure to the change in density. Antuono et al. (2010) proposed to add a density damping term in the continuity equation, given by

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{u} + D \nabla^2 \rho, \quad (\text{A.39})$$

where $D = \delta h c_o$, where $\delta = 0.1$ suggested by Antuono et al. (2010). A convergent discretization of the density damping term proposed by Marrone et al. (2011) is given by

$$\langle \nabla^2 \rho \rangle_i = \sum_j 2\omega_j \left(\frac{(\rho_i - \rho_j)}{r_{ij}} - \mathbf{e}_{ij} \cdot (\langle \nabla \rho \rangle_i + \langle \nabla \rho \rangle_j) \right) \mathbf{e}_{ij} \cdot \nabla W_{ij}, \quad (\text{A.40})$$

where $\langle \nabla \rho \rangle_i$ is the corrected gradient formulation as shown in eq. (1.27) for pressure. The rest of the operators are discretized as in the case of the standard scheme.

A.5.3 Transport Velocity Formulation (TVF)

In the standard and δ -SPH scheme, the particles tend to clump in the presence of negative pressure. Adami et al. (2013) proposed to add a background pressure p_b in the domain to regularize the particle distribution. The particles are advected with the new transport velocity $\tilde{\mathbf{u}}$. Therefore, the modified velocity transport equation is given by

$$\frac{d\tilde{\mathbf{u}}}{dt} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \nabla \cdot (\mathbf{u} \otimes (\tilde{\mathbf{u}} - \mathbf{u})), \quad (\text{A.41})$$

where $\frac{d(\bullet)}{dt} = \frac{\partial(\bullet)}{\partial t} + \tilde{\mathbf{u}} \cdot \nabla(\bullet)$. In TVF, the continuity equation is replaced by the summation density formulation, given by

$$\rho_i = \sum_j m_j W_{ij}, \quad (\text{A.42})$$

and the pressure is computed using a linear equation of state given by

$$p_i = c_o^2(\rho_i - \rho_o). \quad (\text{A.43})$$

Therefore, the particle distribution affects the pressure in the flow. The momentum equation in eq. (A.41) is discretized as

$$\frac{d\tilde{\mathbf{u}}_i}{dt} = \frac{1}{m_i} \sum_j (\omega_i^2 + \omega_j^2) \left[-\tilde{p}_{ij} \nabla W_{ij} + \frac{1}{2} (\mathbf{A}_i + \mathbf{A}_j) \cdot \nabla W_{ij} + \tilde{\mu}_{ij} \frac{\mathbf{u}_{ij}}{r_{ij}} \mathbf{e}_{ij} \nabla W_{ij} \right], \quad (\text{A.44})$$

where

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

$$\tilde{\mu}_{ij} = \frac{2\mu_i \mu_j}{\mu_i + \mu_j},$$

where $\mu = \rho\nu$, and $\mathbf{A} = \rho \mathbf{u} \otimes (\tilde{\mathbf{u}} - \mathbf{u})$. The transport velocity is computed by adding an additional acceleration due to a constant background pressure p_b , given by

$$\tilde{\mathbf{u}}_i = \mathbf{u}_i - \Delta t \frac{p_b}{\rho_i} \sum_j \nabla W_{ij} \omega_j. \quad (\text{A.45})$$

This method generates a more homogenous particle distribution. However, it is not applicable to free surface flow problems since eq. (A.42) is used to evaluate density which underestimated density near the free surface. Also, the normalization form would blow up the particles.

Zhang et al. (2017) proposed Generalized TVF (GTVF) scheme, where the density is evolved using the modified continuity equation, given by

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \tilde{\mathbf{u}} \quad (\text{A.46})$$

They suggested using a variable background pressure that eliminates the tensile instability completely. It allows the scheme to be applicable to free-surface flow and solid material problems. The density is reinitialized after every timestep to avoid large errors at a high Reynolds number.

A.5.4 Entropically Damped Artificial Compressibility (EDAC) scheme

In an attempt to remove the usage of the artificial equation of state, Ramachandran et al. (2019) extended the Entropically Damped Artificial Compressibility (EDAC) formulation proposed by Clausen (2013) to SPH. The pressure is evolved using the pressure evolution equation given by

$$\frac{dp}{dt} = -\rho c_o^2 \nabla \cdot \mathbf{u} + \tilde{\nu} \nabla^2 p, \quad (\text{A.47})$$

where $\tilde{\nu} = \alpha h c_o / 8$, where the recommended value of $\alpha = 0.5$. Equation (A.47) is similar to the density evolution in eq. (A.40); however, the SPH discretization is performed differently, given by

$$\frac{dp_i}{dt} = \sum_j \left[\rho_i c_o^2 \mathbf{u}_{ij} \cdot \nabla W_{ij} + \frac{(\omega_i^2 + \omega_j^2)}{m_i} \tilde{\mu}_{ij} \frac{\tilde{p}_{ij}}{(r_{ij}^2 + \eta h_{ij}^2)} \nabla W_{ij} \cdot \mathbf{x}_{ij} \right], \quad (\text{A.48})$$

where $\tilde{\mu}_{ij} = \frac{2\tilde{\mu}_i\tilde{\mu}_j}{\tilde{\mu}_i+\tilde{\mu}_j}$, where $\mu = \rho\tilde{\nu}$. Ramachandran et al. (2019) also coupled the EDAC formulation with the TVF formulation by incorporating the regularization force in the momentum equation.

A.5.5 Arbitrary Lagrange Eulerian SPH (ALE-SPH) and δ^+ SPH scheme

Recently, Sun et al. (2019) followed by Jacob et al. (2021) and Adepu et al. (2021) suggested improvement to apply the shifting velocity in the governing equations correctly. The original continuity equation in eq. (A.46) is modified to

$$\frac{\tilde{d}\rho}{dt} = -\rho \nabla \cdot \tilde{\mathbf{u}} + \nabla \cdot (\rho \delta \mathbf{u}) + D \nabla^2 \rho, \quad (\text{A.49})$$

where $\delta \mathbf{u} = \tilde{\mathbf{u}} - \mathbf{u}$ is the shifting velocity and D is a controlling parameter as used in the δ -SPH scheme. Similarly, the momentum equation in eq. (A.41) is modified to

$$\frac{\tilde{d}\mathbf{u}}{dt} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \nabla \cdot (\mathbf{u} \otimes \delta \mathbf{u}) - \mathbf{u} \nabla \cdot \delta \mathbf{u}. \quad (\text{A.50})$$

The shifting velocity is evaluated using the particle shifting technique (PST) proposed by Lind et al. (2012), given by

$$\delta \mathbf{u} = -M(2h) c_o \sum_j \left[1 + R \left(\frac{W_{ij}}{W(\Delta s)} \right)^n \right] \nabla W_{ij}. \quad (\text{A.51})$$

The first term in eq. (A.49) is discretized using eq. (1.20) whereas the second term is discretized using eq. (1.21). The density damping term is discretized the same as in the case of the δ -SPH scheme in eq. (A.40). In the eq. (A.50), the pressure gradient term is discretized using eq. (1.24), and the viscous terms are evaluated using the formulation in eq. (1.33). Furthermore, the third term is discretized using the symmetric divergence formulation in eq. (1.21) whereas the last term is discretized using the asymmetric formulation in eq. (1.20).

A.5.6 Eulerian WCSPH scheme

SPH method can be applied to an Eulerian framework as well. Noutcheuwa et al. (2012) followed by Lind et al. (2016) proposed the Eulerian SPH scheme for the incompressible model. Recently, Nasar et al. (2019) proposed Eulerian Weakly-Compressible SPH (EWCSPPH) scheme. The Eulerian NS equation is given by

$$\frac{\partial \rho}{\partial t} = -\rho \nabla \cdot \mathbf{u} - \mathbf{u} \cdot \nabla \rho, \quad (\text{A.52a})$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} - \mathbf{u} \cdot \nabla \mathbf{u}, \quad (\text{A.52b})$$

In the Eulerian framework, particles do not move, and a Cartesian arrangement of particles is used. Therefore, the density ρ of the particle is constant in the domain resulting $\nabla \rho = 0$. Therefore, Nasar et al. (2019) suggests ignoring the last term in the eq. (A.52a). The modified continuity equation is given by

$$\frac{\partial \rho}{\partial t} = -\rho \nabla \cdot \mathbf{u}. \quad (\text{A.53})$$

For all the schemes described above, the time step for the time integrator is given by

$$\Delta t = \min \left(0.3 \frac{h}{U + c_o}, 0.25 \frac{h^2}{\nu} \right), \quad (\text{A.54})$$

where U is the maximum velocity in the domain.

In algorithm 8, we show the generic algorithm for the WCSPH scheme. The inputs for the scheme are the particle arrays with \mathbf{N} number of particles, fluid properties like dynamic viscosity, and initial condition. The scheme produces the results after evolving the properties till the final time $\mathbf{t.f}$. The function `ComputeTimestep` sets the time step for

Algorithm 8: Pseudo-code for WCSPH scheme.

Input: Particle arrays, kinematic viscosity, initial conditions

Result: Evolved properties at $t=tf$

ComputeTimestep();

while $t < tf$ **do**

for $i=1:N$ **do**

ComputePressure();

end

for $i=1:N$ **do**

ComputeDivergence();

ComputeForces();

end

for $i=1:N$ **do**

TakeOneTimeStep();

end

$t = t + dt;$

end

the simulation using the criteria in eq. (A.54). The simulation ends when the time reaches tf . First, the pressure is evaluated for all the particles in **ComputePressure**. This pressure is used to evaluate divergence and forces in **ComputeDivergence** and **ComputeForces**, respectively. These accelerations are used in the integrator in **TakeOneTimeStep**. For brevity, we do not show all the integration stages in the algorithm. Furthermore, we do not consider any boundaries. However, this algorithm can be applied to solve periodic problems like the Taylor-Green vortex and the Gresho-Chan vortex.

A.6 δ^+ SPH formulation correction

The evolution equation of the δ^+ SPH equation has the form

$$\frac{Df}{Dt} = \frac{df}{dt} + \nabla f \cdot \delta \mathbf{u}, \quad (\text{A.55})$$

where $\delta \mathbf{u}$ is the shifting velocity and $\frac{Df}{Dt} = \frac{\partial f}{\partial t} + (\mathbf{u} + \delta \mathbf{u}) \cdot \nabla f$. The above equation for a particle i is given by

$$\frac{Df_i}{Dt} = \frac{df_i}{dt} + \nabla f_i \cdot \delta \mathbf{u}_i. \quad (\text{A.56})$$

We can use the vector identity for the last term given by

$$\nabla f \cdot \delta \mathbf{u} = \nabla \cdot (f \delta \mathbf{u}) - f \nabla \cdot (\delta \mathbf{u}). \quad (\text{A.57})$$

On performing SPH approximation, we obtain

$$\begin{aligned}
\nabla f_i \cdot \delta \mathbf{u}_i &= \sum_j (f_j \delta \mathbf{u}_j - f_i \delta \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j - \\
&\quad \sum_j f_i (\delta \mathbf{u}_j - \delta \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j \\
&= \sum_j (f_j - f_i) \delta \mathbf{u}_j \cdot \nabla W_{ij} \omega_j.
\end{aligned} \tag{A.58}$$

Clearly, we cannot recover the LHS should we use the above discretization. However, on using f_j in place of f_i in the second term, we get

$$\begin{aligned}
\nabla f_i \cdot \delta \mathbf{u}_i &= \sum_j (f_j \delta \mathbf{u}_j - f_i \delta \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j - \\
&\quad \sum_j f_j (\delta \mathbf{u}_j - \delta \mathbf{u}_i) \cdot \nabla W_{ij} \omega_j. \\
&= \sum_j (f_j - f_i) \delta \mathbf{u}_i \cdot \nabla W_{ij} \omega_j.
\end{aligned} \tag{A.59}$$

Thus, in the δ^+ SPH, we should use the discretization in eq. (A.59).

A.7 Schemes with issues solving the Gresho-Chan vortex

In this section, we show the results of the scheme for which the Gresho-Chan vortex problem failed to complete. In fig. A.1, we plot the velocity of the particles with the distance, r from the center at $t = 1.5s$, and the linear momentum in the x-direction with time for a 100×100 simulation. Clearly, all the schemes considered show better approximate conservation of linear momentum compared to other schemes; however, they fail to complete.

In the case of L-RR-C, due to the presence of sharp change in the velocity field, the remeshing procedure diverges (Chaniotis et al., 2002). In the case of E-C, TV-C, and EWCSPH schemes, we suspect that the advection term $\mathbf{u} \cdot \nabla \mathbf{u}$ (or $\delta \mathbf{u} \cdot \nabla \mathbf{u}$ in case of TV-C) diverge in the absence of viscosity. This opens possible avenues of research to obtain a better discretization of the advection term.

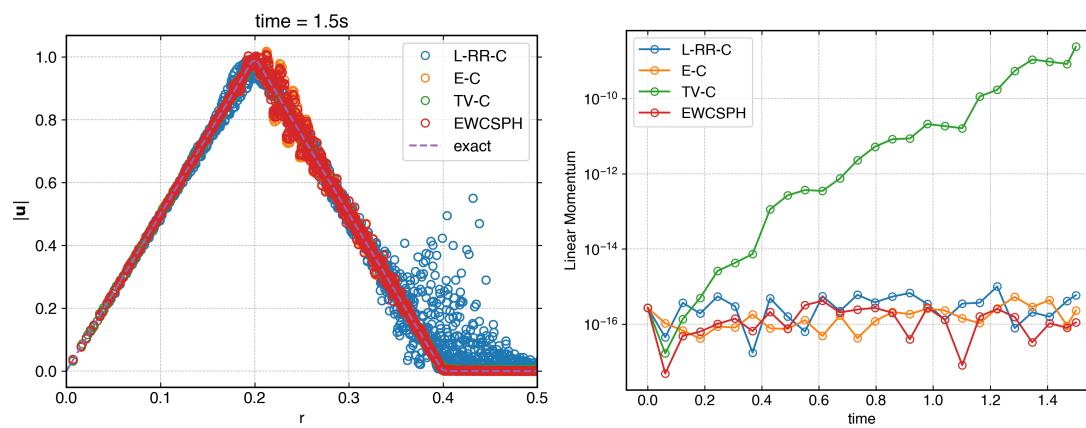


Figure A.1 : The velocity of particles with the distance from the center of the vortex (left) and the x -component of the total linear momentum (right) for the Gresho-Chan vortex problem.

Bibliography

- [1] Adami, S., Hu, X.Y., and Adams, N.A. “A generalized wall boundary condition for smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 231.21 (Aug. 2012), pp. 7057–7075. doi: 10.1016/j.jcp.2012.05.005 (cit. on pp. 124, 125, 159, 161, 163, 164, 166, 180).
- [2] Adami, S., Hu, X.Y., and Adams, N.A. “A transport-velocity formulation for smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 241 (May 2013), pp. 292–307. doi: 10.1016/j.jcp.2013.01.043 (cit. on pp. 11, 13, 30, 38, 41, 107, 205).
- [3] Adepu, Dinesh and Ramachandran, Prabhu. “A corrected transport-velocity formulation for fluid and structural mechanics with SPH”. In: *arXiv preprint arXiv:2106.00756* (2021) (cit. on p. 206).
- [4] Akinci, Nadir, Akinci, Gizem, and Teschner, Matthias. “Versatile surface tension and adhesion for SPH fluids”. In: *ACM Transactions on Graphics (TOG)* 32.6 (2013), pp. 1–8. doi: <https://doi.org/10.1145/2508363.2508395> (cit. on p. 92).
- [5] Antuono, M. 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. 11, 64, 178, 204).
- [6] Antuono, Matteo. “Tri-Periodic Fully Three-Dimensional Analytic Solutions for the Navier–Stokes Equations”. In: *Journal of Fluid Mechanics* 890 (2020). issn: 0022-1120, 1469-7645. doi: 10.1017/jfm.2020.126 (cit. on p. 62).
- [7] Armaly, B et al. “Experimental and Theoretical Investigation of Backward-Facing Step Flow”. In: *Journal of Fluid Mechanics* 127 (Jan. 1983), pp. 473–496. doi: 10.1017/S0022112083002839 (cit. on pp. 130–132).
- [8] Batchelor, George Keith. *An introduction to fluid dynamics*. Cambridge university press, 1967 (cit. on p. 36).

- [9] Bern, Marshall and Eppstein, David. “Mesh generation and optimal triangulation”. In: *Computing in Euclidean geometry 1* (1992), pp. 23–90 (cit. on p. 98).
- [10] Biriukov, Sergei and Price, Daniel J. “Stable Anisotropic Heat Conduction in Smoothed Particle Hydrodynamics”. In: *Monthly Notices of the Royal Astronomical Society* 483.4 (Mar. 2019), pp. 4901–4909. ISSN: 0035-8711. DOI: 10.1093/mnras/sty3413 (cit. on pp. 9, 30).
- [11] Bond, Ryan B et al. “Manufactured solution for computational fluid dynamics boundary condition verification”. In: *AIAA journal* 45.9 (2007), pp. 2224–2236 (cit. on pp. 60, 145).
- [12] Bonet, J. and Lok, T.-S.L. “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. DOI: 10.1016/S0045-7825(99)00051-1 (cit. on pp. 6, 7, 9, 24, 25, 27, 29–31, 33, 34, 44, 55, 186).
- [13] Bonet, Javier and Rodríguez-Paz, Miguel X. “Hamiltonian formulation of the variable-h SPH equations”. In: *Journal of Computational Physics* 209.2 (2005), pp. 541–558. ISSN: 0021-9991. DOI: <https://doi.org/10.1016/j.jcp.2005.03.030> (cit. on p. 2).
- [14] Brookshaw, L. “A Method of Calculating Radiative Heat Diffusion in Particle Simulations”. In: *Publications of the Astronomical Society of Australia* 6.2 (1985), pp. 207–210. ISSN: 1323-3580, 1448-6083. DOI: 10.1017/S1323358000018117 (cit. on pp. 9, 30).
- [15] Chaniotis, A.K., Poulidakos, D., and Koumoutsakos, P. “Remeshed Smoothed Particle Hydrodynamics for the Simulation of Viscous and Heat Conducting Flows”. In: *Journal of Computational Physics* 182.1 (2002), pp. 67–90. ISSN: 0021-9991. DOI: 10.1006/jcph.2002.7152 (cit. on p. 209).
- [16] Chen, J. K. and Beraun, J. E. “A Generalized Smoothed Particle Hydrodynamics Method for Nonlinear Dynamic Problems”. In: *Computer Methods in Applied Mechanics and Engineering* 190.1 (Oct. 2000), pp. 225–239. ISSN: 0045-7825. DOI: 10.1016/S0045-7825(99)00422-3 (cit. on pp. 8, 13, 191).
- [17] Chorin, Alexandre Joel. “A numerical method for solving incompressible viscous flow problems”. In: *Journal of Computational Physics* 2.1 (Aug. 1967), pp. 12–26. ISSN: 0021-9991. DOI: 10.1016/0021-9991(67)90037-X (cit. on pp. 10, 38).

- [18] Choudhary, Aniruddha. “Verification of Compressible and Incompressible Computational Fluid Dynamics Codes and Residual-Based Mesh Adaptation”. Jan. 2015 (cit. on p. 145).
- [19] Choudhary, Aniruddha et al. “Code Verification for Multiphase Flows Using the Method of Manufactured Solutions”. In: *International Journal of Multiphase Flow* 80 (Apr. 2016), pp. 150–163. ISSN: 03019322. DOI: [10 . 1016 / j . ijmultiphaseflow . 2015 . 12 . 006](https://doi.org/10.1016/j.ijmultiphaseflow.2015.12.006) (cit. on p. 65).
- [20] Clausen, Jonathan R. “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. DOI: [10 . 1103 / PhysRevE . 87 . 013309](https://doi.org/10.1103/PhysRevE.87.013309) (cit. on p. 206).
- [21] Cleary, Paul W and Monaghan, Joseph J. “Conduction modelling using smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 148.1 (1999), pp. 227–264. DOI: [10 . 1006 / jcph . 1998 . 6118](https://doi.org/10.1006/jcph.1998.6118) (cit. on pp. 9, 14, 30, 62, 80, 81, 188).
- [22] Colagrossi, Andrea and Landrini, Maurizio. “Numerical Simulation of Interfacial Flows by Smoothed Particle Hydrodynamics”. In: *Journal of Computational Physics* 191.2 (2003), pp. 448–475. ISSN: 0021-9991. DOI: [10 . 1016 / S0021 - 9991 \(03\)00324 - 3](https://doi.org/10.1016/S0021-9991(03)00324-3) (cit. on pp. 124, 125, 159, 161, 163, 164, 166, 180).
- [23] Colagrossi, Andrea et al. “Particle packing algorithm for SPH schemes”. In: *Computer Physics Communications* 183.8 (2012), pp. 1641–1653. DOI: [https : / / doi . org / 10 . 1016 / j . cpc . 2012 . 02 . 032](https://doi.org/10.1016/j.cpc.2012.02.032) (cit. on pp. 18, 70, 89–91, 101, 107, 117).
- [24] Cummins, S. J and Rudman, M. “An SPH projection method”. In: *Journal of Computational Physics* 152 (1999), pp. 584–607 (cit. on p. 10).
- [25] Dehnen, Walter and Aly, Hossam. “Improving convergence in smoothed particle hydrodynamics simulations without pairing instability”. In: *Monthly Notices of the Royal Astronomical Society* 425.2 (2012), pp. 1068–1082. DOI: [10 . 1111 / j . 1365 - 2966 . 2012 . 21439 . x](https://doi.org/10.1111/j.1365-2966.2012.21439.x) (cit. on pp. 13, 19, 56, 100, 186).
- [26] Di, Yana et al. “Moving Mesh Finite Element Methods for the Incompressible Navier–Stokes Equations”. In: *SIAM Journal on Scientific Computing* 26.3 (Jan. 2005), pp. 1036–1056. ISSN: 1064-8275, 1095-7197. DOI: [10 . 1137 / 030600643](https://doi.org/10.1137/030600643) (cit. on pp. 18, 50).

- [27] Dilts, Gary A. “Moving-least-squares-particle hydrodynamics—I. Consistency and stability”. In: *International Journal for Numerical Methods in Engineering* 44.8 (1999), pp. 1115–1155. doi: [https://doi.org/10.1002/\(SICI\)1097-0207\(19990320\)44:8<1115::AID-NME547>3.0.CO;2-L](https://doi.org/10.1002/(SICI)1097-0207(19990320)44:8<1115::AID-NME547>3.0.CO;2-L) (cit. on pp. 8, 27, 28, 33).
- [28] Esmaili Sikarudi, M. A. and Nikseresht, A. H. “Neumann and Robin Boundary Conditions for Heat Conduction Modeling Using Smoothed Particle Hydrodynamics”. In: *Computer Physics Communications* 198 (2016), pp. 1–11. ISSN: 0010-4655. doi: [10.1016/j.cpc.2015.07.004](https://doi.org/10.1016/j.cpc.2015.07.004) (cit. on pp. 124, 164, 166).
- [29] Fatehi, R. and Manzari, M.T. “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](https://doi.org/10.1016/j.camwa.2010.11.028) (cit. on pp. 5, 7, 9, 14, 29–31, 34, 188–190, 199, 201).
- [30] Federico, I. et al. “Simulating 2D Open-Channel Flows through an SPH Model”. In: *European Journal of Mechanics - B/Fluids* 34 (2012), pp. 35–46. ISSN: 09977546. doi: [10.1016/j.euromechflu.2012.02.002](https://doi.org/10.1016/j.euromechflu.2012.02.002) (cit. on p. 126).
- [31] Feng, Hualong et al. “Smoothed Particle Hydrodynamics Simulation of Viscoelastic Flows with the Slip-Link Model”. In: *Molecular Systems Design & Engineering* 1.1 (2016), pp. 99–108. doi: [10.1039/C5ME00009B](https://doi.org/10.1039/C5ME00009B) (cit. on p. 60).
- [32] Ferrand, M. et al. “Unified Semi-Analytical Wall Boundary Conditions for Inviscid, Laminar or Turbulent Flows in the Meshless SPH Method”. In: *International Journal for Numerical Methods in Fluids* 71.4 (2013), pp. 446–472. ISSN: 02712091. doi: [10.1002/flid.3666](https://doi.org/10.1002/flid.3666) (cit. on p. 119).
- [33] Ferrari, Angela et al. “A New 3D Parallel SPH Scheme for Free Surface Flows”. In: *Computers & Fluids* 38.6 (2009), pp. 1203–1217. ISSN: 0045-7930. doi: [10.1016/j.compfluid.2008.11.012](https://doi.org/10.1016/j.compfluid.2008.11.012) (cit. on p. 121).
- [34] Fourtakas, Georgios, Vacondio, Renato, and Rogers, Benedict D. “On the Approximate Zeroth and First-Order Consistency in the Presence of 2-D Irregular Boundaries in SPH Obtained by the Virtual Boundary Particle Methods”. In: *International Journal for Numerical Methods in Fluids* 78.8 (2015), pp. 475–501. ISSN: 02712091. doi: [10.1002/flid.4026](https://doi.org/10.1002/flid.4026) (cit. on p. 121).
- [35] Fourtakas, Georgios et al. “Local Uniform Stencil (LUST) Boundary Condition for Arbitrary 3-D Boundaries in Parallel Smoothed Particle Hydrodynamics

- (SPH) Models”. In: *Computers & Fluids* 190 (2019), pp. 346–361. issn: 0045-7930. doi: 10.1016/j.compfluid.2019.06.009 (cit. on pp. 121, 157, 159).
- [36] Frontiere, Nicholas, Raskin, Cody D., and Owen, J. Michael. “CRKSPH - A Conservative Reproducing Kernel Smoothed Particle Hydrodynamics Scheme”. In: *Journal of Computational Physics* 332.1 (Mar. 2017), pp. 160–209. doi: 10.1016/j.jcp.2016.12.004 (cit. on pp. 8, 13, 28, 29).
- [37] Ghia, U., Ghia, K. N., and Shin, C. T. “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 p. 62).
- [38] Gingold, R. A. and Monaghan, J. J. “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 pp. 1, 2).
- [39] Gresho, Philip M and Chan, Stevens T. “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 (1990), pp. 621–659. doi: <https://doi.org/10.1002/flid.1650110510> (cit. on pp. 18, 48).
- [40] Guerrero, Joel. “Numerical simulation of the unsteady aerodynamics of flapping flight”. In: *Department of Civil, Environmental, Architectural Engineering* (2009) (cit. on pp. 134, 135).
- [41] Gurtin, Morton E. *An introduction to continuum mechanics*. Academic press, 1982 (cit. on p. 187).
- [42] Hashemi, M. R., Fatehi, R., and Manzari, M. T. “A modified SPH method for simulating motion of rigid bodies in Newtonian fluid flows”. In: *International Journal of Non-Linear Mechanics* 47.6 (July 2012), pp. 626–638. issn: 0020-7462. doi: 10.1016/j.ijnonlinmec.2011.10.007 (cit. on pp. 28, 119, 121, 158, 164, 166).
- [43] Hernquist, Lars and Katz, Neal. “TREESPH-A unification of SPH with the hierarchical tree method”. In: *The Astrophysical Journal Supplement Series* 70 (1989), pp. 419–446. doi: 10.1086/191344 (cit. on p. 12).

- [44] Hieber, S. E. and Koumoutsakos, P. “An Immersed Boundary Method for Smoothed Particle Hydrodynamics of Self-Propelled Swimmers”. In: *Journal of Computational Physics* 227.19 (Oct. 2008), pp. 8636–8654. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2008.06.017 (cit. on p. 39).
- [45] Huang, C. et al. “A kernel gradient-free SPH method with iterative particle shifting technology for modeling low-Reynolds flows around airfoils”. In: *Engineering Analysis with Boundary Elements* 106 (Sept. 2019), pp. 571–587. ISSN: 0955-7997. DOI: 10.1016/j.enganabound.2019.06.010 (cit. on pp. 7, 13, 18, 36, 188).
- [46] Jacob, Bruno et al. “An Arbitrary Lagrangian Eulerian Smoothed Particle Hydrodynamics (ALE-SPH) Method with a Boundary Volume Fraction Formulation for Fluid-Structure Interaction”. In: *Engineering Analysis with Boundary Elements* 128 (2021), pp. 274–289. ISSN: 0955-7997. DOI: 10.1016/j.enganabound.2021.04.006 (cit. on p. 206).
- [47] Jiang, Min et al. “Blue noise sampling using an SPH-based method”. In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), pp. 1–11. DOI: <https://doi.org/10.1145/2816795.2818102> (cit. on pp. 89, 91, 92, 106, 116, 117).
- [48] Kiara, Areti, Hendrickson, Kelli, and Yue, Dick K. P. “SPH for incompressible free-surface flows. Part II: Performance of a modified SPH method”. In: *Computers & Fluids* 86 (2013), pp. 510–536. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2013.07.016 (cit. on p. 13).
- [49] Kiara, Areti, Hendrickson, Kelli, and Yue, Dick K.P. “SPH for incompressible free-surface flows. Part I: Error analysis of the basic assumptions”. In: *Computers & Fluids* 86 (Nov. 2013). ISSN: 00457930. DOI: 10.1016/j.compfluid.2013.05.023 (cit. on p. 12).
- [50] Kinsler, Lawrence E et al. “Fundamentals of acoustics”. In: *Fundamentals of Acoustics, 4th Edition, by Lawrence E. Kinsler, Austin R. Frey, Alan B. Coppens, James V. Sanders, pp. 560. ISBN 0-471-84789-5. Wiley-VCH, December 1999.* (1999), p. 560 (cit. on p. 129).
- [51] Korzilius, S. P., Schilders, W. H. A., and Anthonissen, M. J. H. “An Improved CSPM Approach for Accurate Second-Derivative Approximations with SPH”. In: *Journal of Applied Mathematics and Physics* 05.01 (2017), pp. 168–184. ISSN: 2327-4352, 2327-4379. DOI: 10.4236/jamp.2017.51017 (cit. on pp. 8, 9, 14, 30, 31, 190, 191).

- [52] Lastiwka, Martin, Basa, Mihai, and Quinlan, Nathan J. “Permeable and Non-Reflecting Boundary Conditions in SPH”. In: *International Journal for Numerical Methods in Fluids* 61.7 (2009), pp. 709–724. ISSN: 02712091, 10970363. DOI: 10.1002/flid.1971 (cit. on pp. 126–128, 136).
- [53] Lind, S. J. and Stansby, P. K. “High-order Eulerian incompressible smoothed particle hydrodynamics with transition to Lagrangian free-surface motion”. In: *Journal of Computational Physics* 326 (Dec. 2016), pp. 290–311. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.08.047 (cit. on pp. 13, 14, 207).
- [54] Lind, Steven J 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. DOI: 10.1016/j.jcp.2011.10.027 (cit. on pp. 18, 38, 207).
- [55] Liu, Gui-Rong and Liu, Moubin B. *Smoothed particle hydrodynamics: a meshfree particle method*. World scientific, 2003 (cit. on p. 3).
- [56] Liu, M.B. and Liu, G.R. “Restoring particle consistency in smoothed particle hydrodynamics”. In: *Applied Numerical Mathematics* 56.1 (Jan. 2006), pp. 19–36. ISSN: 01689274. DOI: 10.1016/j.apnum.2005.02.012 (cit. on pp. 7, 8, 27, 34, 55, 123, 187).
- [57] Lucy, L. B. “A numerical approach to testing the fission hypothesis”. In: *The Astronomical Journal* 82.12 (1977), pp. 1013–1024 (cit. on p. 1).
- [58] Macià, Fabricio et al. “A Boundary Integral SPH Formulation: Consistency and Applications to ISPH and WCSPH”. In: *Progress of Theoretical Physics* 128.3 (Sept. 2012), pp. 439–462. ISSN: 0033-068X. DOI: 10.1143/PTP.128.439 (cit. on pp. 9, 13, 14).
- [59] Marongiu, J. C., Leboeuf, F, and Parkinson, E. “Numerical Simulation of the Flow in a Pelton Turbine Using the Meshless Method Smoothed Particle Hydrodynamics: A New Simple Solid Boundary Treatment”. In: *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 221.6 (2007), pp. 849–856. ISSN: 0957-6509, 2041-2967. DOI: 10.1243/09576509JPE465 (cit. on pp. 119, 121, 158).
- [60] Marrone, S. 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](https://doi.org/10.1016/j.cma.2010.12.016) (cit. on pp. 13, 90, 98, 112, 117, 122, 159, 161, 163, 164, 166, 177, 180, 204).
- [61] Marrone, S. 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. doi: <https://doi.org/10.1016/j.jcp.2013.03.011> (cit. on pp. 132, 134, 135).
- [62] Meurer, Aaron et al. “SymPy: symbolic computing in Python”. In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. doi: [10.7717/peerj-cs.103](https://doi.org/10.7717/peerj-cs.103) (cit. on p. 65).
- [63] Monaghan, J. J. “Simulating Free Surface Flows with SPH”. In: *Journal of Computational Physics* 110 (1994), pp. 399–406. doi: [10.1006/jcph.1994.1034](https://doi.org/10.1006/jcph.1994.1034) (cit. on p. 6).
- [64] Monaghan, J. J. “Smoothed Particle Hydrodynamics”. In: *Reports on Progress in Physics* 68 (2005), pp. 1703–1759. doi: [10.1088/0034-4885/68/8/R01](https://doi.org/10.1088/0034-4885/68/8/R01) (cit. on pp. 6, 8, 80).
- [65] Monaghan, J. J. and Lattanzio, J. C. “A Refined Particle Method for Astrophysical Problems”. In: *Astronomy and Astrophysics* 149.1 (1985), pp. 135–143. ISSN: 0004-6361 (cit. on p. 2).
- [66] Monaghan, Joe J. “Smoothed particle hydrodynamics”. In: *Annual review of astronomy and astrophysics* 30 (1992), pp. 543–574 (cit. on p. 6).
- [67] Monaghan, Joseph J and Gingold, Robert A. “Shock simulation by the particle method SPH”. In: *Journal of computational physics* 52.2 (1983), pp. 374–389. doi: [10.1016/0021-9991\(83\)90036-0](https://doi.org/10.1016/0021-9991(83)90036-0) (cit. on p. 6).
- [68] Morris, Joseph P., Fox, Patrick J., and Zhu, Yi. “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 pp. 9, 11, 204).
- [69] Morris, Joseph Peter. “A study of the stability properties of SPH”. In: *arXiv preprint astro-ph/9503124* (1995) (cit. on p. 90).
- [70] Morris, Joseph Peter. *Analysis of smoothed particle hydrodynamics with applications*. Monash University Australia, 1996 (cit. on p. 3).

- [71] Muta, Abhinav and Ramachandran, Prabhu. “Efficient and accurate adaptive resolution for weakly-compressible SPH”. In: *Computer Methods in Applied Mechanics and Engineering* 395 (May 2022), p. 115019. ISSN: 0045-7825. DOI: 10.1016/j.cma.2022.115019 (cit. on pp. 2, 184).
- [72] Muta, Abhinav, Ramachandran, Prabhu, and Negi, Pawan. “An efficient, open source, iterative ISPH scheme”. In: *Computer Physics Communications* (2020), p. 107283. DOI: 10.1016/j.cpc.2020.107283 (cit. on pp. 177, 178).
- [73] Nasar, A.M.A. et al. “Eulerian Weakly Compressible Smoothed Particle Hydrodynamics (SPH) with the Immersed Boundary Method for Thin Slender Bodies”. In: *Journal of Fluids and Structures* 84 (Jan. 2019), pp. 263–282. ISSN: 08899746. DOI: 10.1016/j.jfluidstructs.2018.11.005 (cit. on pp. 11, 14, 40, 42, 207).
- [74] Negi, Pawan and Ramachandran, Prabhu. “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 pp. 97, 225).
- [75] Negi, Pawan and Ramachandran, Prabhu. “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. DOI: 10.1063/5.0072383 (cit. on p. 225).
- [76] Negi, Pawan and Ramachandran, Prabhu. “How to Train Your Solver: Verification of Boundary Conditions for Smoothed Particle Hydrodynamics”. In: *Physics of Fluids* 34.11 (Nov. 2022), p. 117125. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0126234 (cit. on p. 225).
- [77] Negi, Pawan and Ramachandran, Prabhu. “Techniques for Second-Order Convergent Weakly Compressible Smoothed Particle Hydrodynamics Schemes without Boundaries”. In: *Physics of Fluids* 34.8 (2022), p. 087125. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0098352 (cit. on p. 225).
- [78] Negi, Pawan, Ramachandran, Prabhu, and Haftu, Asmelash. “An Improved Non-Reflecting Outlet Boundary Condition for Weakly-Compressible SPH”. In: *Computer Methods in Applied Mechanics and Engineering* 367 (2020), p. 113119. ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113119 (cit. on p. 225).

- [79] Noutcheuwa, Rodrigue Kéou and Owens, Robert G. “A New Incompressible Smoothed Particle Hydrodynamics-Immersed Boundary Method”. In: *International Journal of Numerical Analysis Modeling Series B* 3.2 (2012), pp. 126–167. doi: <https://doi.org/> (cit. on p. 207).
- [80] Nugent, S. and Posch, H. A. “Liquid drops and surface tension with smoothed particle applied mechanics”. In: *Physical Review E* 62.4 (Oct. 2000). Publisher: American Physical Society, pp. 4968–4975. doi: [10.1103/PhysRevE.62.4968](https://doi.org/10.1103/PhysRevE.62.4968) (cit. on pp. 9, 30).
- [81] Oberkampf, William, Blottner, Frederick, and Aeschliman, Daniel. “Methodology for computational fluid dynamics code verification/validation”. In: *Fluid dynamics conference*. 1995, p. 2226 (cit. on p. 11).
- [82] Oberkampf, William L and Roy, Christopher J. *Verification and validation in scientific computing*. Cambridge University Press, 2010 (cit. on p. 64).
- [83] Oger, G. 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. ISSN: 0021-9991. doi: [10.1016/j.jcp.2016.02.039](https://doi.org/10.1016/j.jcp.2016.02.039) (cit. on p. 11).
- [84] Quinlan, N. J., Basa, M., and Lastiwka, M. “Truncation error in mesh-free particle methods”. In: *International Journal for Numerical Methods in Engineering* 66.13 (June 2006), pp. 2064–2085. ISSN: 0029-5981, 1097-0207. doi: [10.1002/rme.1617](https://doi.org/10.1002/rme.1617) (cit. on pp. 4, 12, 13, 29, 198).
- [85] Ramachandran, Prabhu. “automan: A Python-Based Automation Framework for Numerical Computing”. In: *Computing in Science & Engineering* 20.5 (Sept. 2018), pp. 81–97. doi: [10.1109/MCSE.2018.05329818](https://doi.org/10.1109/MCSE.2018.05329818) (cit. on pp. 15, 41, 109, 157, 183).
- [86] Ramachandran, Prabhu and Puri, Kunal. “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](https://doi.org/10.1016/j.compfluid.2018.11.023) (cit. on pp. 11, 13, 41, 178, 206).
- [87] Ramachandran, Prabhu et al. “PySPH: A Python-based Framework for Smoothed Particle Hydrodynamics”. In: *ACM Transactions on Mathematical Software* 47.4 (Sept. 2021), 34:1–34:38. ISSN: 0098-3500. doi: [10.1145/3460773](https://doi.org/10.1145/3460773) (cit. on pp. 15, 41, 107, 109, 157, 183, 225).

- [88] Randles, PW and Libersky, Larry D. “Smoothed particle hydrodynamics: some recent improvements and applications”. In: *Computer methods in applied mechanics and engineering* 139.1-4 (1996), pp. 375–408. doi: 10.1016/S0045-7825(96)01090-0 (cit. on pp. 18, 123, 125, 164, 166).
- [89] Roache, Patrick J. *Verification and validation in computational science and engineering*. Vol. 895. Hermosa Albuquerque, NM, 1998 (cit. on p. 11).
- [90] Rosswog, Stephan. “Boosting the accuracy of SPH techniques: Newtonian and special-relativistic tests”. In: *Monthly Notices of the Royal Astronomical Society* 448.4 (2015), pp. 3628–3664. doi: 10.1093/mnras/stv225 (cit. on pp. 7, 8, 13).
- [91] Roy, Christopher J. “Review of Code and Solution Verification Procedures for Computational Simulation”. In: *Journal of Computational Physics* 205.1 (May 2005), pp. 131–156. issn: 0021-9991. doi: 10.1016/j.jcp.2004.10.036 (cit. on p. 12).
- [92] Salari, Kambiz and Knupp, Patrick. *Code Verification by the Method of Manufactured Solutions*. 2000 (cit. on pp. 59, 64).
- [93] Schwaiger, Hans F. “An Implicit Corrected SPH Formulation for Thermal Diffusion with Linear Free Surface Boundary Conditions”. In: *International Journal for Numerical Methods in Engineering* 75.6 (Aug. 2008), pp. 647–671. issn: 00295981, 10970207. doi: 10.1002/nme.2266 (cit. on pp. 9, 13, 14).
- [94] Sharma, Nidhi and Sengupta, Tapan K. “Vorticity Dynamics of the Three-Dimensional Taylor-Green Vortex Problem”. In: *Physics of Fluids* 31.3 (2019), p. 035106. issn: 1070-6631. doi: 10.1063/1.5083870 (cit. on p. 63).
- [95] Shepard, Donald. “A two-dimensional interpolation function for irregularly-spaced data”. In: *Proceedings of the 1968 23rd ACM national conference*. ACM '68. New York, NY, USA: Association for Computing Machinery, Jan. 1968, pp. 517–524. isbn: 978-1-4503-7486-6. doi: 10.1145/800186.810616 (cit. on p. 121).
- [96] Sun, Peng-Nan, Colagrossi, Andrea, and Zhang, A-Man. “Numerical simulation of the self-propulsive motion of a fishlike swimming foil using the δ -SPH model”. In: *Theoretical and Applied Mechanics Letters* 8.2 (2018), pp. 115–125. doi: <https://doi.org/10.1016/j.taml.2018.02.007> (cit. on pp. 28, 36).

- [97] Sun, PN et al. “The δ plus-SPH model: Simple procedures for a further improvement of the SPH scheme”. In: *Computer Methods in Applied Mechanics and Engineering* 315 (2017), pp. 25–49. doi: 10.1016/j.cma.2016.10.028 (cit. on p. 80).
- [98] Sun, PN et al. “A consistent approach to particle shifting in the δ -Plus-SPH model”. In: *Computer Methods in Applied Mechanics and Engineering* 348 (2019), pp. 912–934. doi: 10.1016/j.cma.2019.01.045 (cit. on pp. 6, 11, 13, 42, 43, 206).
- [99] Swegle, JW, Hicks, DL, and Attaway, SW. “Smoothed particle hydrodynamics stability analysis”. In: *Journal of computational physics* 116.1 (1995), pp. 123–134. doi: <https://doi.org/10.1006/jcph.1995.1010> (cit. on pp. 90, 186).
- [100] Tafuni, A. et al. “A Versatile Algorithm for the Treatment of Open Boundary Conditions in Smoothed Particle Hydrodynamics GPU Models”. In: *Computer Methods in Applied Mechanics and Engineering* 342 (2018), pp. 604–624. ISSN: 00457825. doi: 10.1016/j.cma.2018.08.004 (cit. on pp. 97, 126, 127, 132, 134, 135).
- [101] Takeda, H., Miyama, S. M., and Sekiya, M. “Numerical Simulation of Viscous Flow by Smoothed Particle Hydrodynamics”. In: *Progress of Theoretical Physics* 92.5 (1994), pp. 939–960. ISSN: 0033-068X, 1347-4081. doi: 10.1143/ptp/92.5.939 (cit. on pp. 123, 124, 164, 166).
- [102] Vacondio, Renato et al. “Grand challenges for Smoothed Particle Hydrodynamics numerical schemes”. In: *Computational Particle Mechanics* (Sept. 2020). ISSN: 2196-4386. doi: 10.1007/s40571-020-00354-1 (cit. on pp. 12, 17, 33, 63, 181).
- [103] Violeau, Damien. *Fluid mechanics and the SPH method: theory and applications*. Oxford University Press, 2012. doi: 10.1093/acprof:oso/9780199655526.001.0001 (cit. on p. 6).
- [104] Violeau, Damien and Rogers, Benedict D. “Smoothed particle hydrodynamics (SPH) for free-surface flows: past, present and future”. In: *Journal of Hydraulic Research* 54.1 (2016), pp. 1–26 (cit. on p. 1).
- [105] Waltz, J. et al. “Manufactured Solutions for the Three-Dimensional Euler Equations with Relevance to Inertial Confinement Fusion”. In: *Journal of Computational Physics* 267 (June 2014), pp. 196–209. ISSN: 00219991. doi: 10.1016/j.jcp.2014.02.040 (cit. on p. 60).

- [106] Wendland, Holger. “Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree”. In: *Advances in Computational Mathematics* 4.1 (Dec. 1995), pp. 389–396. ISSN: 1019-7168, 1572-9044. DOI: 10.1007/BF02123482 (cit. on p. 3).
- [107] Xu, Rui, Stansby, Peter, and Laurence, Dominique. “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 p. 101).
- [108] Zhang, Chi., Hu, Xiangyu Y. T., and Adams, Nikolaus A. “A generalized transport-velocity formulation for smoothed particle hydrodynamics”. In: *Journal of Computational Physics* 337 (2017), pp. 216–232. DOI: 10.1016/j.jcp.2017.02.016 (cit. on p. 205).
- [109] Zhang, G. M. and Batra, R. C. “Modified Smoothed Particle Hydrodynamics Method and Its Application to Transient Problems”. In: *Computational Mechanics* 34.2 (July 2004), pp. 137–146. ISSN: 1432-0924. DOI: 10.1007/s00466-004-0561-5 (cit. on pp. 8, 13).
- [110] Zhu, Qirong, Hernquist, Lars, and Li, Yuexing. “Numerical convergence in smoothed particle hydrodynamics”. In: *The Astrophysical Journal* 800.1 (2015), p. 6. DOI: 10.1088/0004-637X/800/1/6 (cit. on pp. 12, 24).

List of publications

The following publications are an outcome of the present work:

1. Negi, Pawan and Ramachandran, Prabhu. “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](https://doi.org/10.1016/j.cpc.2021.108008)
2. Negi, Pawan, Ramachandran, Prabhu, and Haftu, Asmelash. “An Improved Non-Reflecting Outlet Boundary Condition for Weakly-Compressible SPH”. in: *Computer Methods in Applied Mechanics and Engineering* 367 (2020), p. 113119. ISSN: 0045-7825. DOI: [10.1016/j.cma.2020.113119](https://doi.org/10.1016/j.cma.2020.113119)
3. Ramachandran, Prabhu, Bhosale, Aditya, Puri, Kunal, Negi, Pawan, Muta, Abhinav, Dinesh, A., Menon, Dileep, Govind, Rahul, Sanka, Suraj, Sebastian, Amal S., Sen, Ananyo, Kaushik, Rohan, Kumar, Anshuman, Kurapati, Vikas, Patil, Mrinalgouda, Tavker, Deep, Pandey, Pankaj, Kaushik, Chandrashekhar, Dutt, Arkopal, and Agarwal, Arpit. “PySPH: A Python-based Framework for Smoothed Particle Hydrodynamics”. In: *ACM Transactions on Mathematical Software* 47.4 (Sept. 2021), 34:1–34:38. ISSN: 0098-3500. DOI: [10.1145/3460773](https://doi.org/10.1145/3460773)
4. Negi, Pawan and Ramachandran, Prabhu. “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. DOI: [10.1063/5.0072383](https://doi.org/10.1063/5.0072383)
5. Negi, Pawan and Ramachandran, Prabhu. “Techniques for Second-Order Convergent Weakly Compressible Smoothed Particle Hydrodynamics Schemes without Boundaries”. In: *Physics of Fluids* 34.8 (2022), p. 087125. ISSN: 1070-6631, 1089-7666. DOI: [10.1063/5.0098352](https://doi.org/10.1063/5.0098352)
6. Negi, Pawan and Ramachandran, Prabhu. “How to Train Your Solver: Verification of Boundary Conditions for Smoothed Particle Hydrodynamics”. In: *Physics of*

Fluids 34.11 (Nov. 2022), p. 117125. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0126234

Acknowledgements

Words cannot express my gratitude to my Ph.D. supervisor Prof. Prabhu Ramachandran for his invaluable guidance and support. I would like to express my deepest appreciation to the contributors of PySPH and automan open-source tools. I also could not have taken this journey without the feedback I received from my research progress committee over the years.

I also would like to express my deepest gratitude to my colleagues Abhinav Muta and Adepu Dinesh for helping me transition into the lab. I am also thankful to Kumara Edara and Lalit Bhola for the long brain-storming about ideas and doubts. I am also thankful to the system admin Vinay Narasimhan for his prompt support for computational resources. I am extremely grateful to the Aerospace department for providing the computational and research lab facility.

I am deeply indebted to my parents, who have helped me emotionally and trusted my decision to go for Ph.D. I could not have taken this journey without the care and understanding extended by my loving wife. I am also grateful to my brother and sister for keeping me motivated. Many thanks to the Frisbee team at IIT Bombay and the Mumbai ultimate community for helping me get going by playing ultimate frisbee for a refreshing end of every day.

I would also like to extend my sincere thanks to the reviewers of my publications who have helped to improve the quality of my work. I would like to mention Dr. Kadambari Devarajan for wordsmithing the title of one of the publications. I also would like to acknowledge Prof. Krishnendu Haldar to allow me to use his workstation to run some of my simulations.

Pawan Singh Negi

IIT Bombay

17 May 2023