

The Application of Smoothed Particle Hydrodynamics to the Simulation of Multiphase Flows Through Packed Beds

Daniel James Barker

December 2014

Imperial College London

Dept. of Earth Science & Engineering

A thesis submitted in part fulfilment of the requirements
for the degree of Doctor of Philosophy.

Supervised by Dr. Stephen J. Neethling

Abstract

Heap leaching is an important hydrometallurgical minerals processing technique, which involves the irrigation of leaching solution through a packed bed of ore particles. It is a complex process governed by the precise chemistry of the ores and leaching solution, the biological action of bacteria that live on the rocks, and the multiphase fluid flow through and around the rocks. In order to improve the predication of recovery and the performance of the operations, a better understanding of all aspects is vital. Several studies have suggested the importance of the fluid flow and mass transport within the heaps.

The smoothed particle hydrodynamics (SPH) method has a proven track record of simulating saturated flows at very small scales for porous media and is a natural tool for gaining an insight into the flows within packed beds. In this thesis we present developments to the handling of wall boundaries in SPH with an aim towards handling arbitrary geometries and improving accuracy. As well as considering the inclusion of surface tension and contact angles for simulating unsaturated flows. A volume factor correction for curved boundaries is developed that demonstrably improves accuracy of simulations, as well as detailing a ‘boundary integral method’ for including wall boundaries.

The developed SPH simulation code is used to run a variety of unsteady unsaturated simulations through 2D packed beds, including the effects of molecular diffusion. These simulations demonstrate the important roles that saturation and capillarity play in packed beds, with unsaturated flows having enhanced transport coefficients as compared to higher saturations. These simulations also serve to highlight the fact that even at relatively long time scales, from the perspective of the flow, that the transport is not in the simple Fickian regime, which is often used when modelling heaps and other packed beds at a larger scale.

Originality Declaration

I hereby declare that this work is original research undertaken by me and that no part of this thesis has been submitted for consideration towards another degree at this or any other institution, and further, that any work which is not my own has been appropriately referenced.

Daniel James Barker

December 2014

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work

Acknowledgements

I would like to thank my supervisor Dr Stephen Neethling for accepting me as his PhD student and providing helpful discussion and direction throughout. He is perpetually cheerful and always willing to take the time to help. I am very grateful to the current and past members of the froths, foams and leaching research group who have made my time at Imperial College hugely enjoyable and productive. Particular thanks to Pablo Brito-Parada for running FEM simulations for me, Gopalkrishnan Parameswaran for helping with work on the SPH code and to Lewis (Qingyang) Lin. Thanks also to my friends and family. Finally I would like to express my gratitude to Rio Tinto for sponsoring this project under the auspices of their Centre for Advanced Minerals Recovery.

Contents

1	Introduction	1
1.1	Thesis Outline	2
2	Heap Leaching Hydrodynamics	5
2.1	Introduction	5
2.2	Heap Preparation & Operation	5
2.2.1	Ore Processing	5
2.2.2	Construction	6
2.2.3	Leaching Solution Application	6
2.3	Flow within Heaps	7
2.4	Modelling of Heap Leaching	8
2.4.1	Heap Scale Models	9
2.4.2	Micro Scale Models	11
2.4.3	Meso Scale Phenomena	12
2.5	Conclusions	13
3	Smoothed Particle Hydrodynamics	15
3.1	Mathematical Framework	15
3.1.1	Interpolation & Gradient Estimation	15
3.1.2	Convergence & Completeness	19
3.1.3	Symmetrization	21
3.2	The Navier-Stokes Equation	22
3.2.1	SPH Equations of Motion	22
3.2.2	Direct vs. Integrated Density	27
3.3	Surface Tension & Contact Angles	28
3.4	Timestepping	31
3.4.1	Timestep Criteria	32
3.5	Pressure Matching	33
3.6	Smoothing Kernels	34
3.7	Conclusions	37
4	Implementation & Computation	39
4.1	The SPH Code	39
4.1.1	Asymptotic Complexity	39
4.1.2	Linked Cell Grid	40
4.1.3	Boundary Particles	42
4.1.4	Periodicity	42
4.1.5	Data Structures & Code Optimization	43
4.1.6	Parallelisation	44
4.2	Postprocessing	48
4.3	Periodic Packings	48
4.4	Conclusions	51
5	Wall Boundaries	53
5.1	Geometric Objects	55
5.1.1	Curved Boundary Correction	55
5.1.2	Example: Couette Flow	57
5.2	B-Splines Curves	60
5.2.1	Gradients and Curvature	61
5.2.2	Reflecting in B-Spline Curves	62
5.2.3	Concave Corners & Piecewise Curves	64
5.2.4	Placing Boundary Particles	65
5.2.5	Application in Three Dimensions	66

5.3	Examples	67
5.3.1	Throttled Pipe Flow	67
5.3.2	Interface Forces	68
5.4	Boundary Integral Method	70
5.4.1	Kernel Inconsistency	70
5.4.2	Calculating γ	71
5.4.3	Boundary Particles	74
5.4.4	Particle Spacing Factor	76
5.4.5	Physical Boundary Conditions	76
5.5	Results	77
5.5.1	Hydrostatic Case	77
5.5.2	Poisseuille Flow	78
5.6	Gradient Correction Order	78
5.7	Contact Angle and Surface Forces	81
5.7.1	Test Case	81
5.7.2	Colour Gradients	82
5.7.3	Continuum Surface Force	83
5.7.4	Curvature Based Surface Force	86
5.7.5	Contact Points and Contact Angles	86
5.7.6	Method 1: Potential Energy Formulation	87
5.7.7	Method 2: Prescribed Normals	89
5.8	Conclusions	91
6	Numerical Stability & Accuracy	93
6.1	Spurious Velocities	93
6.2	Pressure Induced Locking	95
6.2.1	Shear Flow Example	96
6.2.2	Balance of Forces	101
6.2.3	Background Pressures	102
6.3	Conclusions	103
7	Voronoi Surface Tension	105
7.1	Triangulation Algorithm	105
7.2	Surface Tension	107
7.3	Pressure Discontinuity	110
7.4	Conclusions	113
8	Mass Transport in Packed Beds	115
8.1	Dispersion Mechanisms	115
8.1.1	Péclet Numbers & Dispersivities	117
8.1.2	Seepage Velocity	117
8.2	Measuring Dispersion	118
8.2.1	Propagators	120
8.2.2	Path-lines	120
8.2.3	Ergodicity & Averaging	122
8.3	Simulating Molecular Diffusion	123
8.3.1	Timestep Constraint	124
8.3.2	Validation & Error Analysis	125
8.4	Taylor Dispersion	126
8.4.1	Flow in a Pipe	126
8.4.2	Flow in a Pipe with Constriction	128
8.5	Regular Packing	131
8.5.1	Flow Structures	132
8.5.2	Hysteresis and Initial Conditions	134

8.5.3	Dependence on Saturation	136
8.5.4	Velocity Autocorrelation	145
8.5.5	Flow Channel Width	146
8.6	Effects of Geometry	148
8.6.1	Particle Radius	148
8.6.2	Gravity direction for regular packing	150
8.6.3	Irregular packing	150
8.7	Channelling	151
8.8	Segregated Geometries	154
8.9	Conclusion	157
9	Conclusions & Future Directions	159
9.1	Future Work	161
	Bibliography	164
A	Derivations	171
A.1	Derivation of Expected Speed Up - S_p	171
A.2	Derivation of CSF Model	171
A.3	Derivation of Local Pressure Force	173
A.3.1	General Dimensions	175
A.4	Derivation of Kernel Gradient - $\nabla\chi_i$	177
A.5	Derivation of SPH Gradient Estimate - $\langle\nabla A\rangle$	178
A.6	Derivation of Superficial Velocity Time Dependence - $v_s(t)$	180
A.7	Derivation of Dual Population Dispersion - $D(\tau)$	182
A.8	Derivation of the Green-Kubo Relation	183
A.9	Derivation of Striated Geometry Characteristic Length	184
B	Miscellaneous	187
B.1	Software	187
B.2	Publications	187

List of Figures

2.1	Heap Leaching Schematic	6
2.2	Images of Industrial Heaps	7
2.3	Mobile-Immobile Model Illustration	12
3.1	Gather/Scatter	18
3.2	Convergence of SPH method for 2D Poiseuille Flow.	20
3.3	Modified Kernel and Gradient	20
3.4	Examples of SPH Estimates with Discontinuity	24
3.5	Equilibrium Contact Angle Illustration	28
3.6	Timestep Against Smoothing Length	34
3.7	Quintic Spline Kernel	35
3.8	SPH dam break example	36
4.1	Linked Cell Grid Illustration & Stencils	40
4.2	Speed up & Efficiency	46
4.3	Flood Fill Algorithm Steps	47
4.4	Flood filled domain example.	47
4.5	Isosurface Example	49
4.6	Progress of Periodic Packing Algorithm	50
5.1	SPH Kernel Deficiency & Static Ghost Boundary	54
5.2	Reflected ghost particle boundaries	54
5.3	Curved Boundary Illustration	56
5.4	Curved Boundary Initial Densities	58
5.5	Curved Boundary Final Densities	58
5.6	Couette Flow Analytical/SPH Comparison	59
5.7	Couette Flow Convergence	60
5.8	B-Spline Basis Functions	60
5.9	NURBS Example and Newton-Raphson Convergence	63
5.10	Illustration of point symmetry reflection for concave corners	65
5.11	3D NURBS Surface Example	66
5.12	B-Spline boundary example simulation	67
5.13	Throttled pipe velocity profiles.	69
5.14	NURBS Example with Surface Tension	69
5.15	Illustration of integral of γ	73
5.16	Performance of γ approximation.	74
5.17	Relative Error of Boundary Integral Approximation	75
5.18	Particle positioning and γ near a wall using BIM.	78
5.19	Velocity comparison for BIM and reflected boundaries.	79
5.20	Plane-Poiseuille velocity profile time evolution	79
5.21	Kernel support near a boundary	79
5.22	Geometry of a droplet against a wall.	81
5.23	Colour gradient example.	83
5.24	Surface force comparison for reflected and BIM boundaries.	83
5.25	Contact angle comparison for BIM and reflected boundaries.	84
5.26	Droplet Width vs. θ for BIM and Reflected Boundaries.	85
5.27	Resulting droplet shape for potential energy method.	88
5.28	Width & height of droplet for prescribed normal method.	89
5.29	Prescribed normal spreading particle arrangement.	90
6.1	Truncated kernel & Spurious Velocity illustration.	94
6.2	Droplet Oscillation: x Semi-axis and Mean Velocity	95
6.3	Shear Flow Setup	96

6.4	Shear Flow Velocity Profiles	97
6.5	Shear Flow Domain Velocity Example	98
6.6	Crystalline Structure Example	98
6.7	Particle Position Spatial Correlation	99
6.8	Timestep vs. h	100
6.9	Solution Error vs. hpc_0^2/μ	101
6.10	Background pressure locking example.	103
7.1	Delaunay Triangulation Initial Candidate.	106
7.2	Delaunay triangulation example.	107
7.3	Voronoi surface tension force distribution.	108
7.4	CSF / Voronoi surface tension comparison, equal densities	109
7.5	CSF / Voronoi surface tension comparison, unequal densities	110
7.6	Smoothed Laplace Pressure	111
7.7	Discontinuous Laplace Pressure	112
8.1	3D Periodic Packing Simulation Output	116
8.2	Path Length cf. Shortest Distance	120
8.3	Integrated Flow Paths, 2D Example	122
8.4	<i>A posteriori</i> Simulation of Molecular Diffusion	123
8.5	Molecular diffusion in a quiescent fluid.	125
8.6	Taylor dispersion in 2D Poiseuille Flow	127
8.7	Diffusing tracer pathlines.	129
8.8	Taylor dispersion in a throttled pipe.	129
8.9	Propagator for flow in a constricted pipe.	130
8.10	Regular Packing Simulation, Example Output	132
8.11	Snapshots of flow structures for various saturations.	133
8.12	Snapshot of different ICs at $t = 0$ s	134
8.13	v_s comparison for different ICs	135
8.14	Channeling example	135
8.15	Seepage velocity for initial conditions.	136
8.16	Seepage velocity for different saturations.	137
8.17	Dispersion coefficients against time for a regular packing.	139
8.18	Measured final dispersion coefficients vs. saturation.	141
8.19	Pathlines for flow in a regular 2D packing.	142
8.20	Propagators for regular packing for $\theta = 12.5\%$ and 25.0%	143
8.21	Standard deviation of the velocity over the domain.	144
8.22	Velocity autocorrelation as a function of saturation.	146
8.23	Radial and axial dispersions for varying d_c	147
8.24	$\langle v \rangle$ over Ω for regular packing with $d_c = 4\text{mm}$	148
8.25	Seepage velocity for different particle diameters.	149
8.26	Dispersion coefficient as a function of v_s	149
8.27	Occupation plots for an alternative orientation of a regular packing.	150
8.28	Dispersion and Peclet No. for alternative orientation of a regular packing.	151
8.29	Irregular packing occupations.	152
8.30	Seepage velocities for different irregular packings.	152
8.31	Dispersivities for different irregular packings.	153
8.32	Channelling / Hysteresis Example.	154
8.33	Segregated geometry.	155
8.34	Stratified geometry dispersion and velocity autocorrelations.	156
8.35	Seepage velocity for stratified geometry.	157
B.1	Diamond GUI Example	187

Nomenclature

d - Spatial dimensions (either 2 or 3)	μ - Dynamic viscosity (Pa s)
h - Smoothing length (m)	ν - Kinematic viscosity ($\text{m}^2 \text{s}^{-1}$)
Δx - Particle spacing length (m)	L_α - Domain size in the α direction (m)
s - Smoothing factor	$\mathbf{d}\mathbf{x}\mathbf{p}$ - Periodic displacement function (m)
t - Time (s)	λ_c - Capillary length (m)
Δt - Discretised Timestep (s)	α_{kl} - Surface tension between phases k and l (N m^{-1})
$\mathbf{x}, \mathbf{y}, \mathbf{r}$ - Position vectors (m)	θ - Contact angle
$\mathbf{u}$ - Velocity vector (m s^{-1})	τ - Time interval (s)
$\mathbf{a}$ - Acceleration vector (m s^{-2})	D - Hydrodynamic dispersion coefficient ($\text{m}^2 \text{s}^{-1}$)
W - Smoothing kernel (m^{-d})	$\mathbf{D}$ - Hydrodynamic dispersion tensor ($\text{m}^2 \text{s}^{-1}$)
$\cdot_i$ - Value belonging to particle i	D_m - Molecular diffusion coefficient ($\text{m}^2 \text{s}^{-1}$)
$\mathbf{r}_{ij}$ - Vector from j to i (m)	L - Characteristic length scale
r_{ij} - Distance from i to j (m)	U - Characteristic velocity
$\hat{\mathbf{e}}_{ij}$ - Unit vector from j to i	$\tilde{t}$ - Dimensionless time
Ω - Whole simulation domain	$\tilde{D}$ - Dimensionless dispersion coefficient
$\Omega_{\mathbf{x}}$ - Kernel support around $\mathbf{x}$	θ_s - Saturation (%)
$\partial\Omega$ - The boundary of Ω	ε - Bed voidage (porosity)
C - SPH colour function	$\mathbf{Re}$ - Reynolds number
σ - Number density of SPH particles (m^{-d})	$\mathbf{Bo}$ - Bond number
γ - Kernel correction factor	$\mathbf{Pe}_m$ - (Molecular) Diffusion Péclet number
$\hat{\mathbf{n}}$ - Normal vector	$\mathbf{Pe}_d$ - (Hydronamic) Dispersion Péclet number
β_V - Volume correction factor	
∇ - Gradient operator (vector not.)	
$\nabla_{\mathbf{x}}$ - Gradient with respect to a specific vector	
x^α / x_α - The α component of vector $\mathbf{x}$	
$\frac{\partial}{\partial x^\alpha}$ - Gradient operator (component not.)	
$\mathbf{g}$ - Acceleration due to gravity (m s^{-2})	
ρ - Fluid density (kg m^{-3})	
ρ_0 - Fluid rest density (kg m^{-3})	
ρ^r - Relative density - ρ/ρ_0	
c_s - Speed of sound (m s^{-1})	
P - Pressure (Pa)	
P_0 - Atmospheric pressure (Pa)	
B - Fluid bulk modulus ($\text{kg m}^{-1} \text{s}^{-2}$)	
V - Volume (m^3)	
β - Compressibility (Pa^{-1})	

Chapter 1

Introduction

Currently the world population sits at just over 7 billion people and is increasing fast. This growing population combined with the increasing living standards in countries such as China, India, and the developing world have lead to an all time high in demand for metals, and mineral based goods. This trend will only continue to increase demand over time; according to [57] if global demand increases 1% annually then in fifty years minerals processing plants must produce an extra 60% over today's output level. Faced with declining ore grades this will be a significant challenge.

Minerals in the earth's crust represent a finite and diminishing resource, much like crude oil. As mining operations become ever larger in scale and economic value, high grade ores and large ore bodies are being depleted [7]. It is not economically viable to process the remaining low grade or small ore bodies using traditional minerals processing techniques, such as froth flotation, due to their generally high set up, infrastructure and running costs. Because of this mining companies are looking to lower cost alternatives and improved efficiency. One such alternative is heap leaching.

Heap leaching is a hydrometallurgical, minerals processing technique, which has relatively low capital and operating costs, and requires comparatively little infrastructure. Because of this, it is seen as an attractive alternative for processing low grade, and small, ore bodies. Currently heap leaching accounts for around 20% of world copper production [54], but this number is set only to increase in the future. One drawback, however, of heap leaching is that it usually has a lower recovery¹ than other processes; 40-70% for heap leaching compared to 80-90% for froth flotation [102]. These numbers also indicate another disadvantage of heap leaching; its performance varies widely for reasons which are not fully understood. This difference in performance can tip the balance between a heap operation being economically viable or not. For both of these reasons mining and minerals companies are desirous of increased understanding of the heap leaching process; to allow better prediction of heap performance as well as to develop improvements that will increase overall performance.

Heap leaching is a conceptually simple process but one which is subject to a large number of external and operational factors which govern its performance. These range from the rock and ore chemistry, ambient weather conditions, bacterial colonisation of heaps, to the flow of reagents and solutes within the heap. A discussion of the issues surrounding heap leaching can be found in [chapter 2](#). In this thesis we investigate some of the hydrodynamic aspects involved in the heap leaching process. The flows encountered in a heap are complex, in that they are unsaturated and unsteady. The scales at which the flows occur are neither small enough to be capillary dominated nor large enough to be inertia dominated, therefore we must account for both. Smoothed particle hydrodynamics (SPH) is a promising numerical method in this

¹Recovery is the fraction of valuable metal in an ore which is recovered by the processing procedure.

regard as it has been used to simulated interfacial, inertial flows. However, there are issues with the method which make applying it to geometries with complex boundary shapes challenging, especially for interfacial flows. The standard method for handling boundaries in SPH is to decompose the boundaries into facets (or edges in 2D), but this does not work especially well with certain boundary formulations and leads to containment issues. For this reason in this thesis we investigate the use of geometrically defined boundaries as well as a new boundary formulation with the aim of improving the handling of arbitrary shapes in SPH. Using these improvements to simulate the heap flows we aim to investigate the transport properties within heaps which can be used to inform larger scale continuum type models.

The following describes the structure of the thesis and outlines the contributions made.

1.1 Thesis Outline

[Chapter 2](#) introduces the heap leaching method in more detail, expanding on why the hydrodynamics involved play such a key role and why we are interested in studying it. A literature review details the current understanding of aspects of flow in heaps, including experimental studies and previous modelling efforts, in order to motivate the application of this thesis.

[Chapter 3](#) introduces the SPH method itself and presents in detail the mathematical basis for how it works. Following which the equations of motion, including for surface tension, are presented along with considerations for the accuracy or reliability of the method. This chapter also explains *why* SPH is a useful numerical tool for simulating the flows we are interested in in this thesis.

In [chapter 4](#) the computational issues involved with actually implementing a working SPH code are highlighted. It discusses the design choices which were made and performance issues with the code. The parallel nature of the simulation code is detailed including the scaling of speed-up and efficiency. It also includes the details of how the geometries used for the simulations are generated and shows how the post processing of 3D simulations allows for better visualisation of simulation results.

[Chapter 5](#) deals with the inclusion of wall boundaries into the SPH method. This is a non trivial task and one which has seen much attention in recent years, but clearly one which has an enormous bearing on the kinds of simulations in this thesis. In this chapter we show a new improvement for handling curved boundaries and the computational issues involved with including arbitrary shaped curved boundaries using B-splines. The rest of the chapter concerns a recently detailed method for handling boundaries in SPH. A new derivation for the gradient estimator is presented as well as the novel application of these boundaries to interfacial flows.

[Chapter 6](#) looks at suitable parameter choices for the systems studied herein. It is shown that the choice of parameters can have a deleterious effect on the accuracy of the SPH solutions. An analysis of the resulting error is presented as well as a derivation of why the error has the form it does. The ideas in this chapter inform the choice of parameters for the simulations in

the last results chapter.

A new method for handling surface tension in SPH is presented in [chapter 7](#). This new method is a Voronoi based method, which relies on reconstructing the interface topology using Delaunay triangulation. The method displays better performance than other surface tension models used in SPH simulations and, further, allows one to explicitly handle pressure discontinuities in multiphase simulations.

The actual simulations of flows in idealised heap like geometries are shown in [chapter 8](#). The chapter begins with a discussion of the issues surrounding such flows and how mass transport is quantified through the dispersion coefficient and how this is calculated for our SPH simulations. The inclusion into our simulations, and the effect of, molecular diffusion is detailed. Next, the results of unsaturated simulations in our packed bed geometry are analysed in detail to describe the mass transport properties and mechanisms at play in heaps.

The thesis finishes with [chapter 9](#) which provides a summary of the contributions of the work presented in context and discusses the shortcomings and directions to be taken in the future.

Chapter 2

Heap Leaching Hydrodynamics

2.1 Introduction

Heap leaching is one of a number of processing techniques which fall under the umbrella term ‘leaching’, others include *in situ*-, dump- and vat-leaching. All leaching processes have in common, that the valuable metal is chemically extracted directly from the ore rocks. This is in contrast to froth flotation, for instance, which is a physical separation process.

Heap leaching is so named because crushed ore, or sometimes even ‘run of mine’ ore, is piled into heaps atop a thick impermeable membrane known as a leaching pad. Leaching solution is then irrigated onto the top of the heap, flowing down through the system, eventually reaching the leaching pad where it is collected in drainage ditches. On its path through the heap, the solution reacts with the minerals, pulling the metal into solution. Once this ‘pregnant solution’ has been collected from the drainage ditch it is processed to extract the metal from the solution, a step called solvent extraction / electro-winning (SX/EW). Apart from the actual mining of the ore, the SX/EW process is the largest operating cost associated with a heap leaching operation [55]. The processed solvent can now be reused, not only reducing costs but also water requirements; an important factor given the remote location of many mines. [Figure 2.1](#) illustrates the leaching process with a schematic diagram.

The performance of a heap leaching operation can be affected by a number of factors. Foremost amongst the multitude are ambient weather conditions, ore chemistry, and mass transport within the heap, which is primarily governed by hydrodynamic effects. Although it is accepted that the hydrodynamics of heap flows has a large effect on the recovery it is only relatively recently that much attention has been given to the ways in which flow can affect a heap’s operation. A number of recent studies have looked at flow within a heap leaching context.

2.2 Heap Preparation & Operation

2.2.1 Ore Processing

The construction of the heap and other primary operations can have a large effect on the hydrodynamics displayed by a given heap. Typically the ore used in the heap’s construction has been crushed, and sometimes then agglomerated [16]. These operations are performed to reduce the variability in ore particle size. However, mines sometimes use so-called ‘run of mine’ ore to construct a heap, whereby the mined ore is not treated in anyway before entering the heap. This is less common, though, as it leads to a higher range of particle sizes (polydispersity) in the heap, which is typically thought to reduce the heap’s permeability and thereby hinder the flow of leaching solution [7]. The initial crushing procedure leads to the presence of ‘fines’

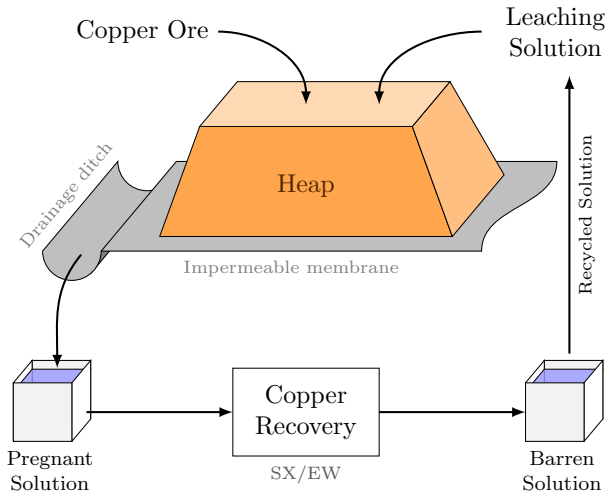


Figure 2.1: A schematic diagram of the heap leaching process.

amongst the ore particles. These fine are particles with sizes $\lesssim 74 \mu\text{m}$. Because the fines are so small they are readily transported by the flow within the heaps and can form impermeable layers which are severely detrimental to leaching performance [75]. The agglomeration process ameliorates this effect by binding the fines and smaller particles to the surface of larger ore particles. After processing the ore particles have a size typically of the order of 2 cm.

2.2.2 Construction

Once processed, the heap is constructed by piling the ore onto the leaching pad using either large dump trucks or by using conveyor belts. The leaching pad is important because it prevents the leaching solution from polluting the local ground water. This is crucial, not only from an environmental perspective, but also an economical one as any loss of leaching solution represents an unrecoverable cost for the mining company. Leaching pads are sometimes reused, with leached ore being moved to a final disposal location after the leaching operation is complete [7]. In other heaps the new ore is piled on the existing heap forming a new layer called a ‘lift’ with each lift being between 3 m and 10 m in height [10].

The construction process, especially when using trucks, can sometimes lead to particle size segregation; a well known phenomena in granular physics, which is sometimes called ‘the Brazil nut effect’ [104]. The segregation of ore particles by size leads to some areas of the heap have very low permeability (small particle size) and others having a much larger permeability (large particle size). This heterogeneity leads to channelling of the fluid (preferential flow) [103].

2.2.3 Leaching Solution Application

The leaching solution is irrigated onto the heap from the top through plastic tubing, which is both robust and resistant to corrosion. The flow rates should be low enough to avoid flooding the heaps since oxygen is a necessity for many of the chemical reactions which take place in



(a) Construction



(b) Aerial view

Figure 2.2: 2.2a A heap being constructed by a conveyor machinery, the highlighted person indicates the scale of an individual lift. 2.2b An aerial view of a heap leaching operation indicating the large scale on which leaching takes place.

the heap [10, 16] as well as being necessary for the survival of bacteria which act to catalyse the reactions [65]. In an unflooded heap the motion of air is driven by the heat of reaction generated inside the heap which causes convection currents to establish themselves, thereby replenishing the oxygen levels. Despite this, in some heaps air is pumped into the bottom of the heap in a process called air sparging in an effort to improve leaching performance [65]. Despite this necessity for air to be present, heaps do sometimes flood due to the presence of fines, particle segregation, or even the precipitation of various minerals which form impermeable layers [35].

2.3 Flow within Heaps

There are two length scales of importance for fluids in heaps. Firstly there is the scale of the interstitial spaces formed between the ore particles, this is typically in the size range of millimetres. This is the scale at which fluid flows around, and between, the ore particles. Secondly the ore particles are porous meaning the fluid can ingress into the rocks themselves.

The length scale of the pores is $\approx 10 \mu\text{m}$, 2 – 3 orders of magnitude smaller than that of the interstitial flow. The large difference in scale leads to very different behaviour for these two flows. A useful metric for capillary flows is the dimensionless Bond number

$$\text{Bo} = \frac{\rho g L^2}{\alpha}, \quad (2.1)$$

where ρ is the fluid density, g is the magnitude of the gravitational acceleration, L is a characteristic length scale and α is the surface tension of the liquid-gas interface. The Bond number is the ratio of forces arising due to gravity to forces caused by capillarity. If $\text{Bo} \gg 1$ then the flow is dominated by gravity, and thus inertial effects, whereas if $\text{Bo} \ll 1$ then the flow is defined far more by capillarity than the influence of gravity. We can see that for the two length scales present in heap leaching the Bond numbers are around 10^{-5} for the flow within the ore particles and $1/5 - 5$ for the interstitial flow.

The very small Bond number (10^{-5}) for intra-particle flow indicates, unsurprisingly, that flow within the rocks is almost entirely capillary dominated at small scales – although at the scale of heaps gravity does play a role in these flows. For the interstitial flows the Bond number straddles unity meaning that *neither* capillarity nor gravity is the dominant effect.

2.4 Modelling of Heap Leaching

Traditionally modelling of heap leaching has focussed on the two distinct scales of the whole heap and the pore scale. Modelling the whole heap is a formidably complex undertaking due to the many factors which are at play in an active heap; fluid flow, chemical reactions and heat and mass transport. The large size of heaps¹ means that directly simulating the fluid flow using the Navier-Stokes equation is completely unfeasible even on today's most powerful supercomputers. Thus, researchers appeal to *continuum models* whereby the actual underlying flow is averaged and described not by Navier-Stokes, but by constitutive equations which govern the evolution of this averaged flow [21, 26, 65, 74, 79]. At the other extreme many studies have examined the flow through porous media, such as the ore rocks, at the pore scale [14, 84, 105]. Typically this involves directly simulating the fluid as it flows through the pore space therefore such studies necessarily have a limited field of view. Between these two scales lies that of the interstitial flows of vital importance to leaching performance. Relatively little modelling effort has been expended at this scale, but there are a number of experimental studies which have focussed here. It is phenomena at this intermediate scale which dominate the flows in heaps over short time scales² and thus should be properly accounted for in the heap scale models, especially where transiencies are present. Thus one aim of this thesis is to inform the continuum models used for heap prediction.

¹Tens of metres in height and thousands of metres to a side.

²Hours to days.

2.4.1 Heap Scale Models

Flow

As discussed above, direct simulation of a whole heap is simply not possible with today's technology, and, even if it were, it would be exceedingly difficult to disentangle the salient information from the overriding complexity. Continuum models attempt to overcome this by considering the average flow properties at each point in the system. The most basic of such continuum models consider only the aqueous phase. The velocity at each point in the system can be obtained by solving Darcy's equation

$$\mathbf{u} = -\frac{K(\hat{\theta})}{\epsilon\mu} \nabla(P + \rho \mathbf{g} \cdot \mathbf{x}), \quad (2.2)$$

combined with the continuity equation

$$\frac{\partial \theta}{\partial t} = -\nabla \cdot \mathbf{u}, \quad (2.3)$$

where K is hydraulic conductivity, $\hat{\theta}$ is the water content³, P the pressure, ρ the fluid density, μ the fluid's dynamic viscosity, $\mathbf{g}$ is the acceleration due to gravity, $\mathbf{x}$ is the position vector and ϵ is the rock porosity. Darcy's equation, however, does not involve time and can only provide the steady state velocity. This is insufficient for heap leaching models because heap flow is unsteady; the heap begins dry before irrigation, heaps can be periodically 'washed' to speed up subsequent reactions and they are subject to external factors such as rainfall. Time dependence can be included by generalizing Darcy's equation to Richard's equation

$$\frac{\partial \hat{\theta}}{\partial t} = \nabla \cdot (K(\hat{\theta}) \nabla P), \quad (2.4)$$

Still another level of sophistication can be added to the models by accounting for both the aqueous and gaseous phases present in the heaps. In this case each point in the system has two velocities, corresponding to the air and the water respectively. The two phases can be coupled together in a number of ways, either by using their pressures, their saturations or a mixed pressure-saturation scheme [23], or only interacting via some intermediate effect. For example, in [65] they assume a constant water velocity, solve the air flow at steady state then have the two interact only via the cooling of the air by the liquid phase.

Many other papers [21, 26, 74] use a more realistic coupling of the two phases. Such a scheme is implemented in, for instance, [79] where they simulated unsaturated flows in heterogeneous packed beds. They were able to obtain good agreement between their simulations and lab-scale experiments for time varying flow rates and saturations which demonstrated hysteresis in the liquid hold-up. Similar unsaturated flow was demonstrated at heap scales in [10, 20, 34] amongst others.

³The fraction of the *total* volume (including solids) occupied by the fluid.

Transport

While some papers focus solely on the flow in the heaps, many attempt to model the transport of reagents and heat on top of the flow [10, 20, 21, 26]. In the majority of papers that show 2D and 3D simulations the transport is governed by the advection-diffusion equation

$$\frac{dC}{dt} = \nabla \cdot (\mathbf{D} \nabla C) - \mathbf{u} \cdot \nabla C, \quad (2.5)$$

where C is the concentration of some substance, and $\mathbf{D}$ is the *dispersion tensor*. The dispersion tensor is a second rank tensor⁴, which takes the role of the diffusion constant in the heat equation. The dispersion tensor accounts for the fact that fluid flow does not spread substances equally in different directions relative to the flow. Substances are mixed more in the direction parallel to the flow than perpendicular to the flow. It is defined as

$$\mathbf{D} = (\alpha_T |\mathbf{u}| + D_m) \mathbf{I} + \frac{(\alpha_L - \alpha_T)}{|\mathbf{u}|} \mathbf{u} \otimes \mathbf{u}. \quad (2.6)$$

where α_T and α_L are the transverse and longitudinal dispersivities, D_m is the molecular diffusion coefficient, $\mathbf{u}$ is the flow velocity of the fluid phase and $\mathbf{I}$ is the identity tensor. In [21] they ignore the effects of molecular diffusion and consider transport to be purely due to hydrodynamic effects, a not unreasonable assumption given $\alpha_T, \alpha_L \gg D_m/|\mathbf{u}|$. In their 2D model they take the transverse dispersivity α_T to be approximately 40% of the longitudinal dispersivity α_L ; $\alpha_L = 1.704 \times 10^{-2}$ m and $\alpha_T = 0.637 \times 10^{-2}$ m.

The advection-diffusion equation is useful for simulating transport because it is well understood and it is relatively easy to include in numerical solvers. Additionally, in the long time limit hydrodynamic dispersion must eventually display Fickian (that is, diffusive) behaviour⁵ [32]. However many authors have argued that this is not a very reasonable assumption over short timescales [12, 97], although mostly in the arena of porous media, which operates at very different length scales (and thus Bond numbers) to the interstitial flows of heap leaching. This will be investigated further in chapter 8.

Attempts have been made to capture the non-Fickian behaviour at short times in larger scale models, with several methods having been suggested. Tompson & Gray published a series of papers [97–99] in which they developed, and then applied, a so-called ‘second order’ model to describe the dispersion behaviour. In these they consider the dispersive flux field to be a variable of the system which is fed in to form the dispersion tensor describing transport. To close the extended set of equations they must, however, introduce another set of constitutive relations, each of which contains more parameters. They showed that this extended equation set could (with suitable parameter values) recover the short time non-Fickian behaviour well for a model 1D system.

Another technique for capturing this behaviour, which has seen some attention in recent years, is the use of fractional diffusion equations. Fractional diffusion equations generalize the advection-

⁴Really a tensor field if it is spatially varying.

⁵Diffusive transport means that $\text{VAR}(C) \propto t$.

diffusion equation to fractional derivative orders (in both time and space). Because the fractional derivative operations are non-local they are capable of capturing the memory effects which lead to the transport in porous flows being (generally) super-diffusive⁶. This technique has the added benefit that the fractional derivative and integral operators are, like their integer counterparts, linear meaning they can be simulated using traditional numerical techniques such as finite difference, albeit with differing stencil weights. A recent review by Benson et al [11] provides a nice overview of how super diffusive behaviour arises from the fractional diffusion equation and numerical techniques for simulating it. Like other methods such fractional diffusion equations still require parametrization and have the added issue of choosing exactly which fractional order is appropriate. Despite their apparent promise such techniques have yet to gain a foothold for simulating large scale flows in multiple dimensions. This is possibly because the non-local nature of the equations means they are computationally expensive to solve.

2.4.2 Micro Scale Models

The flow within the ore particles themselves falls in the area of flow in porous media. The two main areas of research for such flows are oil & gas recovery and hydrogeological areas such as groundwater contamination. Oil and gas recovery typically involves active pumping or large natural pressure gradients, whereas in a heap the only driving force is gravity. This means that the sorts of flows encountered in the oil and gas literature are subject to much higher pressure gradients than the flows seen in heaps. Hydrogeological applications are much closer to heaps in this regard. There have been a few studies at the micro/pore scale whose stated applications were for heap leaching. Such as Pereira et al [86], for instance, where smoothed particle hydrodynamics (SPH) was used to study the drag coefficients of flows through porous media of varying permeabilities. They found a good agreement between their saturated SPH simulations and the theoretical expected results suggesting SPH can be a useful tool for such simulations.

Other studies have also used SPH to study various aspects of flow in porous media. Ovaysi & Piri [84] used SPH to study the velocity fields and pressure drops in X-ray CT images of porous rocks, but the field of view in this study is very small, being only $(1\text{mm})^3$. Also at this scale, Sousa & Jiang [92] simulated flow through a number of 2D periodic geometries, thereby confirming that SPH can accurately recreate Darcy's law (see [equation \(2.2\)](#)). Even though they do not study dispersion, in their paper they also highlight another benefit of SPH for such studies; it does not suffer from numerical diffusion, a common issue with explicit treatments of advection. This fact is also nicely demonstrated in Zhu & Fox [105] where they study dispersion in saturated porous flow using SPH. Even for long times they found that without the addition of molecular diffusion to their simulations the measured dispersion was still Taylor⁷ as one would expect. A detailed discussion of this is presented in [section 8.4](#).

⁶Super-diffusive transport means that $\text{VAR}(C) \propto t^a$ with $a > 1$

⁷Taylor transport means that $\text{VAR}(C) \propto t^2$ and arises in systems with no mixing or diffusion.

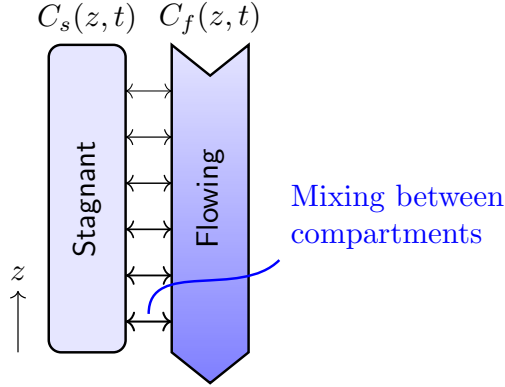


Figure 2.3: The mobile-immobile model considers the fluid phase being in two ‘compartments’ only one of which is flowing. The fluid mixes between the two depending on the values of θ_s and θ_f at a position.

2.4.3 Meso Scale Phenomena

Between the pore scale direct simulation and the heap scale continuum models sits the meso scale flow between the ore particles. There are relatively few simulation papers at this scale, but this is counterbalanced by a number of experimental studies with modelling elements.

In recent years several authors have performed a series of laboratory scale leaching experiments in columns [15, 29, 51]. All measured the residence time distribution using a solvent tracer, from their measurements they demonstrate that stagnant fluid which is held up at capillary junctions has a large effect on the flow. They fitted the experimental results to different mathematical models. The best fit was found to a so-called mobile-immobile model. This is an example of a compartment model, where the fluid phase is subdivided into two compartments – a stagnant part and a flowing part. At each point in the system we have two concentrations of any solute, C_s and C_f for the stagnant and flowing compartments respectively. Transfer between the two compartments can be added to the governing advection-diffusion equation yielding

$$\frac{\partial C_f}{\partial t} = \nabla \cdot (\mathbf{D} \nabla C_f) - \mathbf{u} \cdot \nabla C_f - \frac{k_m}{\epsilon \phi} (C_f - C_s), \quad (2.7)$$

$$\frac{\partial C_s}{\partial t} = -\frac{k_m}{\epsilon(1-\phi)} (C_s - C_f). \quad (2.8)$$

where the constant k_m governs the mass transfer between the phases and ϕ is the fraction of the fluid in the flowing compartment. The model exhibits a long tail off of tracer concentration after the initial (approximately) Gaussian plume has exited the system. Given that the studies by Ilankoon et al [51] and de Andrade Lima [29] used non porous glass beads in their columns this deviation from advection-dispersion must be a feature of the interstitial flow and not due to the effects of particle porosity.

The fits to data were good in these studies but in the conclusions of [29] the author points out that the experimental set up is not capable of capturing the preferential flow channeling, accumulation of fines and other issues seen in real heaps. Additionally, small laboratory scale

columns can be significantly affected by wall effects. Indeed it is the conclusion of Eriksson and Destouni [35] that preferential flow is the main cause of poor leaching performance seen in waste rock heaps in northern Sweden. Wu et al. [103] investigated the effect on metal sulphide grains in a preferential flow channel but, despite emphasizing the importance of preferential flow, did not compare leaching efficacy or effects between this and a non-preferential flow path.

Another effect which is commonly seen in systems such as packed beds and trickle bed reactors is hysteresis; the observed behaviour depends on the flow rate history. This clearly has implications for the operation of a heap. For trickle bed reactors the hysteresis behaviour is well known and has been studied in depth ([73] and references therein). Trickle bed reactors are similar to the packed beds found in heap leaching but they operate with a wide variety of liquid and gas flow rates, thus setting them apart from the flows seen in heap leaching operations. Hysteresis as seen in packed beds was the main focus of I.M.S.K Ilankoon's PhD thesis [53] in which he confirmed, through experimental studies, that the main mechanism driving hysteresis is the formation of flow rivulets. Once formed, the rivulets are stable under reduction of flow rates. This is as opposed to the other possible mechanisms of changes in the shape of, or path taken by, the rivulets. Although it can be included through capillary pressure terms [79], equations such as Richards equation (equation (2.4)) depend only on the current state of the system and so do not capture the experimentally observed hysteresis.

2.5 Conclusions

The flows which are present in packed-beds and, in particular, heap leaching operations are very complicated, being as they are multiphase and subject to a variety of effects such as capillarity, heap construction and temperature. The performance of the recovery is seen to be highly dependent on the flow within the heaps. Because of this many authors have focused on trying to model the fluid flow and transport within whole heaps. However, there is somewhat of a disconnect between the well studied microscopic pore scale flows and the large heap scale flows which are modelled using continuum type models.

The pore scale phenomena have been studied in detail by many authors using a range of experimental and numerical techniques. In the last decade several papers have been published which have demonstrated that SPH is a suitable method for simulating these pore scale flows and that it has desirable properties for extracting information from these simulations. While the intermediate scale interstitial flow has received experimental, and some modelling, attention there have been few papers simulating flow at this scale. The desirable properties of SPH make this an area ripe for exploration. Therefore one aim of this thesis is to investigate the transport phenomena at this intermediate scale using SPH.

Chapter 3

Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a computational and numerical technique used to simulate fluid flow and other related continuum mechanics type problems. It was developed initially by Lucy [72] and Gingold & Monaghan [40] for the study of stellar formation and proto-planetary disks but has since been applied and adapted to a wide variety of problems. SPH is both meshless and Lagrangian; meaning there is no prescribed topology between the integration points (SPH particles) and that the SPH particles move with the fluid flow implicitly handling advection.

These factors give the benefit that in SPH any interfaces between fluid phases are automatically carried with the flow. Often, when accounting for advection terms explicitly one encounters spurious ‘numerical diffusion’ but SPH does not suffer from this issue. Further, no specialized interface tracking is required. This means that the complex topological changes in interface caused by rivulet formation, droplet formation and coalescence are *automatically* handled by the SPH method. Additionally, since each SPH particle represents a parcel of fluid it is possible to formulate SPH to be exactly conservative with respect to mass. These two factors are the primary reasons for the choice of SPH to simulate the flows seen in heap leaching; it provides a robust numerical method for capturing the unsteady interfacial flows that are seen in real heaps.

Much of the work in this thesis relies upon the underpinnings of the SPH method. For this reason a thorough introduction to the mathematical framework is presented in this chapter.

3.1 Mathematical Framework

SPH is used to simulate continuum problems over some domain $\Omega \subset \mathbb{R}^d$ in d dimensions. Depending on the problem at hand one is interested in a number of scalar and vector fields over Ω . In the following we will take $\phi(\mathbf{x})$ to be some scalar field and $\mathbf{A}(\mathbf{x})$ to be some vector field, with $\mathbf{x} \in \Omega$.

3.1.1 Interpolation & Gradient Estimation

As with any discretization scheme, given the value of some scalar or vector field at a specific set of locations (in this case the locations of the SPH particles) we seek a way to interpolate the known values to the whole simulation domain Ω as well as calculating the relevant gradients. The values of any functions of interest are known at the SPH particle positions and are denoted $\{\phi_i = \phi(\mathbf{x}_i)\}$ and $\{\mathbf{A}_i = \mathbf{A}(\mathbf{x}_i)\}$, where $\{\mathbf{x}_i : i = 1 \dots N\}$ are the SPH particle locations.

Table 3.1: Table listing properties that an SPH smoothing kernel must have.

Constraint		
1	$\int_{\Omega_{\mathbf{x}}} W(\mathbf{x} - \mathbf{y} , h) d\mathbf{y} = 1$	Normalization
2	$W(r, h) = W(-r, h)$	Symmetry
3	$W(r, h) = 0$ for $r \geq 2h$	Locality

The standard SPH interpolation function is given by

$$\langle \phi \rangle(\mathbf{x}) = \sum_{i=1}^N \phi_i W(|\mathbf{x} - \mathbf{x}_i|, h) V_i, \quad (3.1)$$

where $V_i = 1/\Delta x^d$ is the volume that the particle i is occupying and $W(\cdot, h)$ is a weighting function. This equation simply says that given some values at specific locations $\mathbf{x}_i$, we take a weighted average of all the function values at each particle based on the particle's distance from $\mathbf{x}$. Equation (3.1) is a discretization of the smoothing operation

$$\langle \phi \rangle(\mathbf{x}) = \int_{\Omega} \phi(\mathbf{y}) W(|\mathbf{x} - \mathbf{y}|, h) d\mathbf{y}, \quad (3.2)$$

thus W is usually referred to as the *smoothing kernel*.

For both the smoothing operation and the interpolation to make sense, W must meet certain criteria, which are listed in table 3.1. The second criterion, symmetry, can alternatively be viewed as requiring a zero mean;

$$\int (\mathbf{x} - \mathbf{y}) W(|\mathbf{x} - \mathbf{y}|, h) d\mathbf{y} = \mathbf{0}. \quad (3.3)$$

The criteria of zero mean is in fact more general than the symmetry constraint as written in table 3.1. It is possible to use an elliptical (ellipsoidal in 3D) smoothing kernel to better capture anisotropic effects [85]. In this case instead of a single smoothing length (or scalar smoothing length field) one has a smoothing length tensor (field).

The third is introduced for computational reasons (see section 4.1.1). The region of finite support around a point $\mathbf{x}$ is given the symbol $\Omega_{\mathbf{x}}$. Taken together these constraints imply that $\lim_{h \rightarrow 0} W(r, h) = \delta(r)$ from which it follows that for equation (3.2)

$$\lim_{h \rightarrow 0} \langle \phi \rangle(\mathbf{x}) = \phi(\mathbf{x}).$$

While we do not necessarily need to interpolate values to arbitrary points to perform a simulation (since we know the values at the SPH particles), it is nevertheless useful since it allows remeshing of quantities to structured points if required (c.f. section 4.2).

In order to numerically solve PDEs one also needs to be able to calculate gradients of functions. With structured (i.e. not meshless) techniques this is done using finite difference or shape functions, for example. In SPH instead a form similar to equation (3.1) is used to calculate

gradients. To derive it, consider [equation \(3.2\)](#) applied to the gradient of ϕ ;

$$\langle \nabla \phi \rangle(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} W(|\mathbf{x} - \mathbf{y}|, h) \nabla \phi(\mathbf{y}) \, d\mathbf{y}. \quad (3.4)$$

This can be split using the product rule into

$$\langle \nabla \phi \rangle(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} \nabla_{\mathbf{y}} [\phi(\mathbf{y}) W(|\mathbf{x} - \mathbf{y}|, h)] - \phi(\mathbf{y}) \nabla_{\mathbf{y}} W(|\mathbf{x} - \mathbf{y}|, h) \, d\mathbf{y}, \quad (3.5)$$

the first term of which becomes a surface integral by applying the divergence theorem;

$$\langle \nabla \phi \rangle(\mathbf{x}) = \oint_{\partial\Omega_{\mathbf{x}}} \phi(\mathbf{y}_S) W(|\mathbf{x} - \mathbf{y}_S|, h) \hat{\mathbf{n}}_S \, dS - \int_{\Omega_{\mathbf{x}}} \phi(\mathbf{y}) \nabla_{\mathbf{y}} W(|\mathbf{x} - \mathbf{y}|, h) \, d\mathbf{y}. \quad (3.6)$$

By noticing that W is zero for all points on the boundary $\partial\Omega_{\mathbf{x}}$ we can see the first integral is identically zero and that by $\nabla_{\mathbf{y}} W(|\mathbf{x} - \mathbf{y}|, h) = -\nabla_{\mathbf{x}} W(|\mathbf{x} - \mathbf{y}|, h)$ we can swap the sign of the second integral to obtain

$$\langle \nabla \phi \rangle(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} \phi(\mathbf{y}) \nabla_{\mathbf{x}} W(|\mathbf{x} - \mathbf{y}|, h) \, d\mathbf{y}. \quad (3.7)$$

This can then be discretized as

$$\langle \nabla \phi \rangle(\mathbf{x}) = \sum_{i=1}^N \phi(\mathbf{x}_i) \nabla_{\mathbf{x}} W(|\mathbf{x} - \mathbf{x}_i|, h) V_i. \quad (3.8)$$

Note how the gradient has been transferred onto the smoothing kernel. Because W is known, so is ∇W making [equation \(3.8\)](#) suitable for estimating a function's gradient given its values at the SPH particles. This result is the linchpin of the SPH method.

Similar results hold for dot products and cross products of vector fields¹;

$$\langle \nabla \cdot \mathbf{A} \rangle(\mathbf{x}) = \sum_{i=1}^N \mathbf{A}_i \cdot \nabla_{\mathbf{x}} W(|\mathbf{x} - \mathbf{x}_i|, h) V_i, \quad (3.9)$$

$$\langle \nabla \times \mathbf{A} \rangle(\mathbf{x}) = \sum_{i=1}^N \mathbf{A}_i \times \nabla_{\mathbf{x}} W(|\mathbf{x} - \mathbf{x}_i|, h) V_i. \quad (3.10)$$

$$(3.11)$$

The gradient of the smoothing kernel can also be written

$$\nabla_{\mathbf{x}} W(|\mathbf{x} - \mathbf{y}|, h) = \frac{\partial}{\partial r} \left(W(r, h) \right) \hat{\mathbf{e}}, \quad (3.12)$$

where $r = |\mathbf{x} - \mathbf{y}|$ is the distance between $\mathbf{x}$ and $\mathbf{y}$ and $\hat{\mathbf{e}}$ is a unit vector pointing from $\mathbf{y}$ towards $\mathbf{x}$. This form is more useful for performing computations.

¹The derivations are practically identical to that shown above.

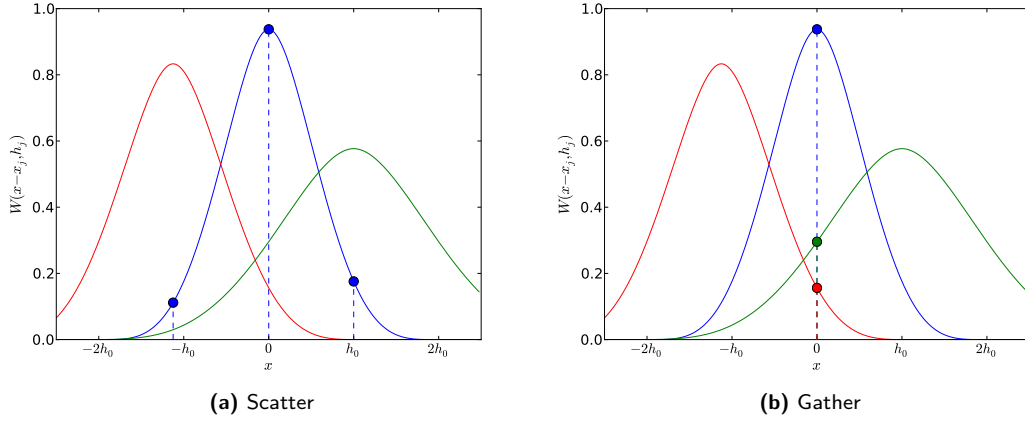


Figure 3.1: Illustration of the scatter and gather formulations for a spatially varying smoothing length in one dimension.

One can also derive similar expressions for the scalar and vector Laplacian operations;

$$\langle \nabla^2 \phi \rangle(\mathbf{x}) = \sum_{i=1}^N \phi_i \nabla_{\mathbf{x}}^2 W(|\mathbf{x} - \mathbf{y}|, h), \quad (3.13)$$

but these are seldom used in practice because they suffer from numerical issues (see [section 3.2.1](#)) meaning the estimates they produce are inaccurate.

Gather/Scatter Formulation

Being completely general, the smoothing length h can vary as a function of both space and time $h = h(\mathbf{x}, t)$. This leads to two different possible choices of SPH smoothing function. The *scatter* form

$$\langle \phi \rangle(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} \phi(\mathbf{y}) W(|\mathbf{x} - \mathbf{y}|, h(\mathbf{x}, t)) d\mathbf{y},$$

and the *gather* form

$$\langle \phi \rangle(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} \phi(\mathbf{y}) W(|\mathbf{x} - \mathbf{y}|, h(\mathbf{y}, t)) d\mathbf{y}.$$

They are so called because the scatter formulation involves taking the value of $\mathbf{x}$'s kernel at the positions $\mathbf{y}$, whereas the gather form involves taking the value of the $\mathbf{y}$'s kernels at $\mathbf{x}$. These two are illustrated in [figure 3.1](#).

Variable smoothing lengths are very useful for problems with structures of interest which are very different in scale, such as the original astrophysical simulations for which SPH was developed [40, 58, 72] or free surface flows where one is especially interested in the interface [36, 82]. The systems studied in this thesis are not suitable, however, for variable smoothing length methods since there are no large volumes where resolution can be reduced, thus we will only consider h constant in both space and time. In this case the two formulations become degenerate and we do not make a choice between one or the other; thus for brevity we drop the

parameter h when writing W henceforth.

3.1.2 Convergence & Completeness

Of primary importance for any numerical method is the order of convergence, both in space and time. With SPH there are two sources of error for spatial convergence. The smoothing step, which has error $O(h^2)$ and the discretization step, which has error

$$\varepsilon \sim O\left(h^{-n} \left(\frac{\Delta x}{h}\right)^2\right) \quad (3.14)$$

for the n^{th} derivative [9]. According to (3.14) the discretization error always decreases quicker than $O(h^2)$ (for $n > 1$) and so the SPH method displays $O(h^2)$ spatial convergence. To see that the smoothing step has an error $O(h^2)$ consider the Taylor expansion of $\phi(\mathbf{y})$ around a point $\mathbf{x}$

$$\phi(\mathbf{y}) = \phi(\mathbf{x}) + \sum_{\alpha=1}^d \frac{\partial \phi}{\partial x_{\alpha}} \Big|_{\mathbf{x}} \cdot (y_{\alpha} - x_{\alpha}) + R((\mathbf{y} - \mathbf{x})^2).$$

Substituting this into the smoothing function equation (3.2) gives

$$\begin{aligned} \langle \phi \rangle(\mathbf{x}) &= \int_{\Omega_{\mathbf{x}}} \left[\phi(\mathbf{x}) + \sum_{\alpha=1}^d \frac{\partial \phi}{\partial x_{\alpha}} \Big|_{\mathbf{x}} \cdot (y_{\alpha} - x_{\alpha}) + R((\mathbf{y} - \mathbf{x})^2) \right] W(|\mathbf{x} - \mathbf{y}|, h) d\mathbf{y} \\ &= \phi(\mathbf{x}) \int_{\Omega_{\mathbf{x}}} W(|\mathbf{x} - \mathbf{y}|, h) d\mathbf{y} + \sum_{\alpha=1}^d \frac{\partial \phi}{\partial x_{\alpha}} \Big|_{\mathbf{x}} \int_{\Omega_{\mathbf{x}}} (y_{\alpha} - x_{\alpha}) W(|\mathbf{x} - \mathbf{y}|, h) d\mathbf{y} + R(h^2) \end{aligned}$$

Since the kernel is normalized the first term is simply $\phi(\mathbf{x})$, the integral in the second term is odd (since $W(r, h)$ is even)² meaning it integrates to zero and we are left with a remainder depending on h^2 , thus

$$\langle u \rangle(\mathbf{x}) = u(\mathbf{x}) + O(h^2).$$

Figure 3.2 shows the convergence of our SPH code for the analytically solvable case of a flow in a 2D pipe (also known as Poiseuille flow, see section 5.5.2) using the Wendland quintic spline smoothing function (see section 3.6). For a smoothing factor ($s = h/\Delta x$) of $s > 1.3$ we see the expected second order convergence but interestingly for a smoothing factor of 1.3 (a common choice in the SPH literature [3, 76]) the convergence appears to not even be first order. This is most likely due to the interplay of the regular particle arrangement in this simple test case and the form of the smoothing kernel used. It has been shown that SPH is surprisingly sensitive to the exact form of the smoothing kernel and the arrangement of the particles [22, 61].

The concept of consistency³ can be hard to work with in meshless methods. Therefore most authors consider completeness [9, 69, 70]. The completeness of a particular method is defined to be the order of polynomial function which it is capable of exactly reproducing⁴. The stan-

²This is the zero mean constraint from above.

³A numerical method is consistent if $\forall \epsilon > 0 \exists$ a discretization length $h(\epsilon)$ s.t. the discretization error $< \epsilon$.

⁴The order of a polynomial function is defined as the order of the highest derivative which is not zero somewhere in the domain.

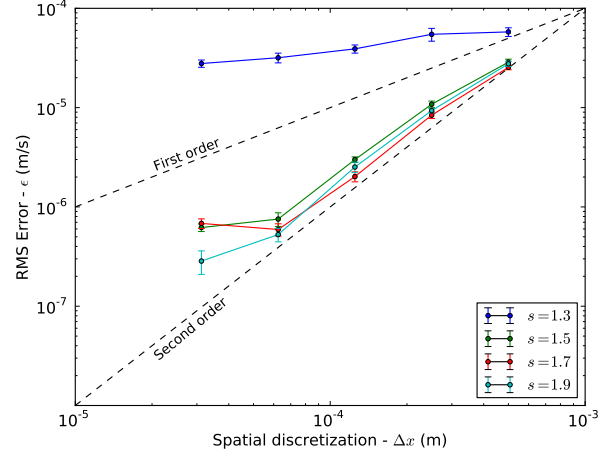


Figure 3.2: The RMS error for 2D Poiseuille flow plotted as a function of the spatial discretization length Δx for different smoothing factors s .

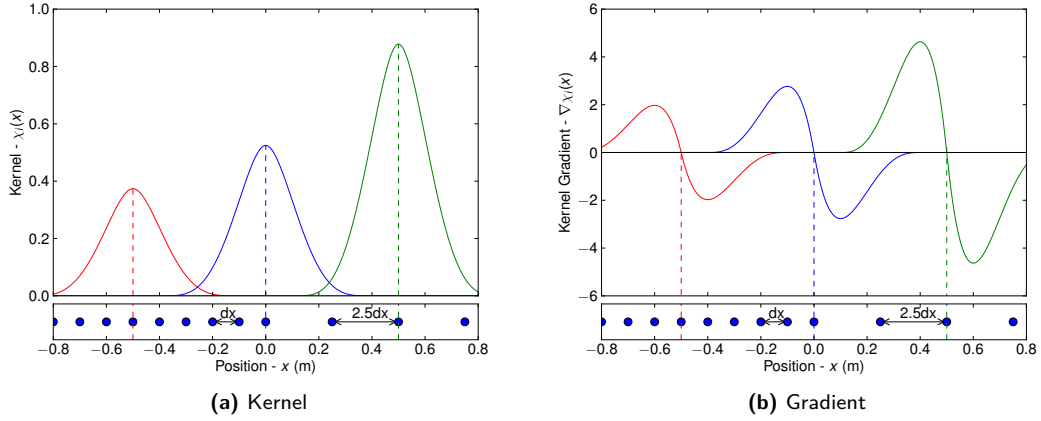


Figure 3.3: Zero order complete smoothing kernel and its spatial gradient in the region near a change in particle spacing. The particle spacing is exaggerated here for illustrative purposes.

dard SPH interpolation function (3.1) is *not even zero order complete*, i.e. it cannot exactly reproduce a constant function. To see this consider a constant function $C(\mathbf{x}) = C$ and its SPH interpolation function

$$\langle C \rangle(\mathbf{x}) = \sum_{j=1}^N W(|\mathbf{x}_j - \mathbf{x}|) C \neq C. \quad (3.15)$$

The reason that this does not exactly reproduce C is that for an arbitrary placement of particles the sum of the smoothing kernel only *approximately* equals unity. Thus $\langle \phi \rangle(\mathbf{x}_i) \neq \phi_i$ and the function (3.1) is only an approximate interpolation.

Particle Inconsistency

The failure of a particular SPH interpolation to reproduce functions of a given order due to particle spacing is called *particle inconsistency*. A variant of the standard SPH interpolation which is zero order complete is

$$\langle \phi \rangle(\mathbf{x}) = \sum_{i=1}^N \phi_i \chi_i(\mathbf{x}), \quad (3.16)$$

where

$$\chi_i(\mathbf{x}) = \frac{W(|\mathbf{x} - \mathbf{x}_i|)}{\sum_{j=1}^N W(|\mathbf{x} - \mathbf{x}_j|)}, \quad (3.17)$$

and following [49] we define the denominator to be $\sigma(\mathbf{x}) = \sum_{j=1}^N W(|\mathbf{x} - \mathbf{x}_j|)$. The factor of σ^{-1} normalizes the discrete sum over particle weights thereby ensuring zero order completeness;

$$\begin{aligned} \langle C \rangle(\mathbf{x}) &= \sum_j \chi_j(\mathbf{x}) C, \\ &= \sum_j \frac{W(|\mathbf{x}_j - \mathbf{x}|)}{\sum_k W(|\mathbf{x}_k - \mathbf{x}|)} C, \\ &= \frac{\sum_j W(|\mathbf{x}_j - \mathbf{x}|)}{\sum_k W(|\mathbf{x}_k - \mathbf{x}|)} C, \\ &= C. \end{aligned}$$

Since σ_i represents the number density of SPH particles around particle i we can see also that $\sigma_i^{-1} \approx V_i$, though it is important to note that whilst SPH particles have a volume, the actual region of space they represent is not well defined. Equation (3.16) thus repairs particle inconsistency up to zeroth order. This modified kernel was originally developed by Shepard [90] and has been widely applied to SPH [39, 48, 49, 89].

3.1.3 Symmetrization

An important feature of any numerical scheme is conservation of quantities which are conserved in reality. When applying the SPH formalism above to a physical system as in the next section, we must ensure conservation of momentum. The SPH scheme will be momentum conserving if the force caused by a particle i on particle j is equal and opposite to the force felt by i due to j . The form of the gradient given by equation (3.8) is *not* conservative because it is not anti-symmetric under interchange of indices i and j . A simple *ad hoc* way to make equation (3.8) symmetric used in [80, 81] is to take the average of the quantities ϕ_i and ϕ_j

$$\langle \nabla \phi \rangle(\mathbf{x}_i) = \sum_j \left(\frac{\phi_i + \phi_j}{2} \right) \nabla W_{ij} V_j.$$

This is, however, incorrect as it will actually underestimate any gradients by a factor of 1/2. The correct form, which is used in many recent SPH papers, is

$$\langle \nabla \phi \rangle(\mathbf{x}_i) = \sum_j (\phi_i + \phi_j) \nabla W_{ij} V_j. \quad (3.18)$$

This can be derived easily. The product rule tells us $\nabla \phi = 1 \cdot \nabla \phi + \phi \cdot \nabla 1$. Now taking the SPH estimates of the gradients on both sides yields

$$\langle \nabla \phi \rangle(\mathbf{x}_i) = 1 \cdot \sum_j \phi_j \nabla W_{ij} V_j + \phi_i \sum_j 1 \cdot \nabla W_{ij} V_j.$$

The sums can be combined to give exactly [equation \(3.18\)](#) which yields anti-symmetric forces (the factor of V_j accounts for the fact this is a force density). For the zero-order complete kernel used here and elsewhere, a similar derivation⁵ gives

$$\langle \nabla \phi \rangle(\mathbf{x}_i) = \sigma_i \sum_j \left(\frac{\phi_i}{\sigma_i^2} + \frac{\phi_j}{\sigma_j^2} \right) \nabla W_{ij} \quad (3.19)$$

as the antisymmetric gradient function, where σ is the SPH particle number density from above.

3.2 The Navier-Stokes Equation

In its Lagrangian form the Navier-Stokes equation for an incompressible Newtonian fluid is

$$\rho \frac{\partial \mathbf{u}}{\partial t} = -\nabla P + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \quad (3.20)$$

where $\mathbf{u}$ is the fluid velocity field, t is time, P the pressure, μ the dynamic viscosity, and $\mathbf{f}$ any addition body forces such as gravity or surface tension.

The Navier-Stokes equation is really a set of d equations (one for each component of $\mathbf{u}$) but it has $d + 1$ unknowns (the d components of $\mathbf{u}$ plus P). To close the system of equations the incompressible continuity equation

$$\nabla \cdot \mathbf{u} = 0, \quad (3.21)$$

is applied. Sometimes also known as the divergence free criteria, [equation \(3.21\)](#) says that we are not creating or destroying fluid volume at any point the system. For an incompressible fluid this is equivalent to mass conservation since a given volume of fluid always has the same mass.

3.2.1 SPH Equations of Motion

The left hand side of the Navier-Stokes equation is the force felt by an infinitesimal fluid parcel, or effectively a *force density*. The simulation domain Ω is decomposed into a set of small but

⁵See [appendix A.4](#).

finite ‘fluid elements’ each of which is represented by an SPH particle. Each SPH particle i therefore has a mass m_i and volume V_i .

Since the volume element is small we can assume the shear over the element itself is small and that the fluid parcel moves coherently as one unit. By moving the particles according to the correct equations of motion, the physical quantities are carried along with it automatically accounting for advection. Thus from [equation \(3.20\)](#) we can write the acceleration of each particle as

$$\mathbf{a}_i = \frac{1}{\rho_i} \left(-\nabla P_i + \mu \nabla^2 \mathbf{u}_i + \mathbf{f}_i \right). \quad (3.22)$$

Clearly we need to calculate the values of ρ_i , $-\nabla P_i$, $\mu \nabla^2 \mathbf{u}_i$ and $\mathbf{f}_i$ for each particle in order to update the particle positions.

The derivation of the relevant forces and accelerations presented here is based on heuristic requirements of simulating the Navier-Stokes equation, [\(3.20\)](#), accurately. It is possible to obtain the same equations using a variational approach as detailed in a review article by Springel [\[93\]](#). Since the focus of [\[93\]](#) is astrophysics they begin with the case of inviscid flow, which is appropriate for their purposes, but viscosity can be included through modification of the Lagrangian.

Density

The density for particle i can be written directly using [equation \(3.1\)](#). Since $V_i = \frac{m_i}{\rho_i}$ it is typical to write the SPH approximations with $\frac{m_i}{\rho_i}$ replacing V_i which gives

$$\rho_i = \sum_j m_j W_{ij}. \quad (3.23)$$

However this suffers from issues at fluid interfaces where there are large density steps. With an air/water system, for instance, the density ratio is ≈ 840 at $298K$. Because of the smoothing nature of the interpolation, the estimated density no longer has a sharp step meaning we will seriously overestimate the density in the lighter phase, and similarly underestimate it in the heavier phase when we are near the boundary. This effect can be seen illustrated in [figure 3.4](#). This over- and underestimation of density leads to a large pressure gradient which causes spurious forces on the interface particles [\[24\]](#). Instead we use the number density $\sigma_i \approx V_i^{-1}$ to give

$$\rho_i = m_i \sum_j W_{ij}. \quad (3.24)$$

This form only references i ’s mass and thus we avoid problems at interfaces as shown in [figure 3.4](#).

It is normally specified that each particle has a fixed mass meaning that as V_i changes so must ρ_i . Although other choices are also valid (see [\[66\]](#)) fixed mass is the usual choice because it enforces exact conservation of mass. This varying ρ means it is possible to simulate compressible fluids, which, depending on the application, may be a desirable or un-desirable feature. Truly

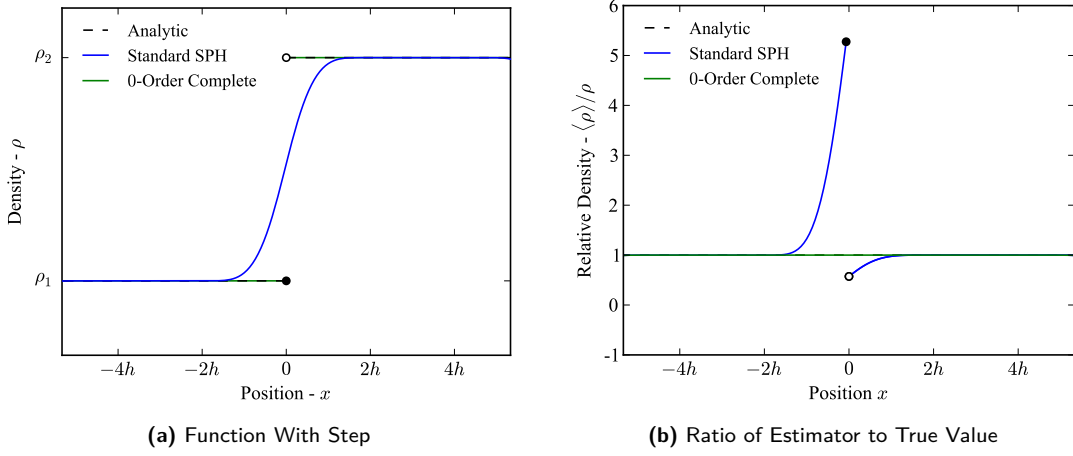


Figure 3.4: Various SPH interpolants of a function with a sharp step. Such discontinuities arise in the density between different fluid phases. We can see how standard SPH smooths out the density.

incompressible formulations of SPH have been developed [59, 67, 84, 94].

Pressure

In order to estimate the pressure gradient we must first calculate the pressure. The simplest way to calculate the pressure is to calculate it from an equation of state such as the ideal gas law $PV = Nk_B T$ from which can write an equation of state for the SPH pressure

$$P = c_0^2 \rho. \quad (3.25)$$

This simple equation of state follows the behaviour of ideal gases and causes the fluid simply to expand without end or until it has filled some bounded volume. In some papers [47] an additional offset pressure P_0 is added to improve numerical stability;

$$P = P_0 + c_0^2 \rho. \quad (3.26)$$

Some authors [33] include a rest density ρ_0

$$P = c_0^2 (\rho - \rho_0). \quad (3.27)$$

This form of the equation captures the essence of intermolecular forces present in fluids which keep the molecules in the liquid phase. If the density decreases below ρ_0 the pressure becomes negative which acts to pull the fluid elements back together.

Since equations (3.25), (3.27) and (3.26) are only linear in density, large density fluctuations can arise when using these. When simulating (nearly) incompressible fluids a different equation

of state often used is the so-called Tait equation;

$$P = B \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right). \quad (3.28)$$

where B is related to the bulk modulus of the fluid and γ is usually taken in the range 1-7 and specifies how harshly to penalise density fluctuations. Becker et al. [8] include a discussion of the Tait equation in which they detail how to choose B so as to obtain density fluctuations of the desired order, namely

$$B = \frac{\rho_0 v_{\max}^2}{\gamma \eta} \quad (3.29)$$

where $\eta = |\Delta\rho|/\rho$ is the desired relative density fluctuation and $v_{\max}$ is the expected maximum speed at any point in the fluid. They point out that this necessitates smaller time-steps to achieve a high degree of near-incompressibility than a truly incompressible method. However the larger number of smaller time-steps is outweighed by the ease and speed with which one can compute the pressure, i.e. without implicitly solving the Poisson equation. In [chapter 6](#) we examine the effects of the equation of state on the performance of the SPH method in detail.

Pressure Gradient

Now we have the pressure from an explicit equation-of-state we can compute its gradient. We use the symmetric estimator (3.19) and write

$$-\nabla P_i = -\sigma_i \sum_j \left(\frac{P_i}{\sigma_i^2} + \frac{P_j}{\sigma_j^2} \right) \nabla W(r_{ij}). \quad (3.30)$$

Viscous Stress

The viscous stress term for an incompressible Newtonian fluid is given by $\mu \nabla^2 \mathbf{u}$. Since SPH estimates for Laplacians are very noisy and offer poor estimation of the true value it is common to combine a finite difference approximation with the SPH estimate for the gradient instead. The form used by several papers [61, 80] is

$$\mu \nabla^2 \mathbf{u}_i = \mu \sum_j \frac{m_j}{\rho_j} (\mathbf{u}_j - \mathbf{u}_i) \nabla^2 W(r_{ij}). \quad (3.31)$$

This form is valid because in a Lagrangian frame such as used here one is only interested in velocity differences. As with the pressure gradient we use the form (3.19) combined with a finite difference approximation giving

$$\mu \nabla^2 \mathbf{u}_i = \mu \sigma_i \sum_j \frac{\mathbf{u}_{ij}}{x_{ij}} \left(\frac{1}{\sigma_i^2} + \frac{1}{\sigma_j^2} \right) \frac{dW_{ij}}{dx_{ij}}, \quad (3.32)$$

where $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ is the difference in velocity between particles i and j . To see where this equation comes from consider the (vector) Laplacian of the velocity field $\nabla^2 \mathbf{u}$. We know that $\nabla^2 \mathbf{u} = \nabla \cdot (\nabla \mathbf{u})$. This can be approximated with the SPH divergence estimate of the velocity gradient tensor, i.e.

$$\nabla \cdot (\nabla \mathbf{u}) \approx \sum_j \nabla W_{ij} \cdot \nabla \mathbf{u}, \quad (3.33)$$

but

$$\nabla W_{ij} = \hat{\mathbf{e}}_{ij} \frac{dW_{ij}}{dr_{ij}}, \quad (3.34)$$

where $\hat{\mathbf{e}}_{ij}$ is a unit vector from particle i to particle j . This gives

$$\nabla \cdot (\nabla \mathbf{u}) \approx \sum_j \hat{\mathbf{e}}_{ij} \cdot \nabla \mathbf{u} \frac{dW_{ij}}{dr_{ij}}. \quad (3.35)$$

physically the vector $\hat{\mathbf{e}}_{ij} \cdot \nabla \mathbf{u}$ can be thought of as the rate of change of the velocity vector in the direction $\hat{\mathbf{e}}_{ij}$ which can clearly be approximated as

$$\hat{\mathbf{e}}_{ij} \cdot \nabla \mathbf{u} \approx \frac{\mathbf{u}_{ij}}{r_{ij}}, \quad (3.36)$$

Combining this with [equation \(3.35\)](#) and symmetrizing yields [equation \(3.32\)](#).

Both [equation \(3.31\)](#) and [equation \(3.32\)](#) have the desirable property that for two particles moving with exactly the same velocity the viscous stress is necessarily zero, because $\mathbf{u}_i - \mathbf{u}_j = 0$. Depending on the choice of $W_h(r)$ the viscous forces in this form can suffer from a numerical instability known as ‘tensile instability’.

Tensile Instability

Tensile instability is a well known artefact in SPH implementations. It is closely related to the choice of smoothing kernel. Swegle et al [95] have shown that a sufficient condition for growth of the instability is

$$H[W]_{\alpha\alpha} (\nabla \mathbf{u})^{\alpha\alpha} > 0 \quad (3.37)$$

where $H[W]_{\alpha\alpha}$ is second derivative of W wrt x_α , i.e. the diagonal of the Hessian matrix. Intuitively the cause can be seen by noticing that the derivatives of the kernels are generally not positive everywhere this means that at certain particle spacings the stresses and pressures can act in a direction opposite to the physically realistic direction.

It has been suggested that specialized smoothing kernels can alleviate this effect to some extent [78]. Others have suggested the problem in essence boils down to a lack of sampling needed to accurately compute $\nabla^2 \mathbf{u}$ so propose a method whereby the usual SPH particles are supplemented by separate ‘stress particles’ (see [70] and references therein for a discussion).

The approach we take, as explained in [76], is to use an artificial pressure term which applies a repellent force on two particles when they come close to one another. For two particles i and j

the value R is defined such that

$$R = \begin{cases} \epsilon \left(\frac{|P_i|}{\rho_i^2} + \frac{|P_j|}{\rho_j^2} \right) & \text{if } P_i < 0 \text{ or } P_j < 0 \\ 0.01 \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) & \text{otherwise} \end{cases},$$

then an additional repulsive force is added between the particles with a magnitude of

$$R \left(\frac{W(r_{ij})}{W(\Delta x)} \right)^n. \quad (3.38)$$

The strength of this force is controlled by the exponent n and the factor ϵ . The values suggested by Monaghan are $\epsilon = 0.2$ and $n = 4$, in which case the force is sufficient to improve the particle spacing without degrading the accuracy of the SPH method.

3.2.2 Direct vs. Integrated Density

Direct calculation of the densities from particles masses at each time-step using [equation \(3.24\)](#) is called *direct density*; it is a common choice for density calculations since it necessarily satisfies [equation \(3.21\)](#). However it is not always possible. If the system contains a free surface then the density will be underestimated at the fluid's edge because of a lack of particles in the kernel's support. A solution to this problem is to make use of the continuity equation. The compressible continuity equation relates the density change to the velocity field;

$$\frac{\partial \rho_i}{\partial t} = -(\nabla \cdot \mathbf{u})_i. \quad (3.39)$$

A finite difference/SPH approximation for this equation is

$$\frac{\rho_i^{(t+1)} - \rho_i^{(t)}}{\Delta t} = - \sum_j m_j (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{x}_{ij}). \quad (3.40)$$

By rearranging this we get a formula for ρ_i at the next time step in terms of the known current density and velocities. This allows us to update the SPH particles' densities based on the fluid motion, this is called *integrated density*.

Integrated density can save a loop over j because we use the current density $\rho_i^{(t)}$ when calculating the other quantities so we can simply calculate the right hand side of [equation \(3.40\)](#) as we calculate all other quantities. However as we calculate σ_i anyway, unless there is a free surface then direct density is usually the better choice. The reason for this is the scheme no longer exactly conserves mass if we update the densities by integrating the continuity equation; numerical errors build and the densities wander from their true values. To overcome this the density can be periodically reinitialised which stops the value becoming too different from the

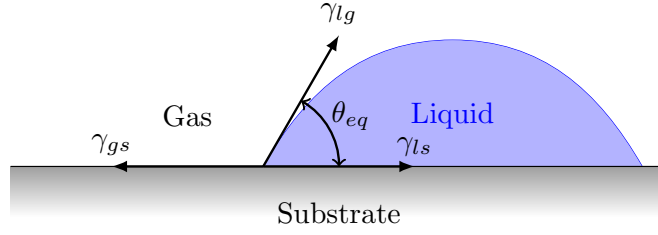


Figure 3.5: Illustration of how the balance of surface energies between the three phases balance at the thermodynamic equilibrium contact angle.

correct value [24]. Each particle i has its density updated as

$$\rho_i = \frac{\sum_j W_{ij} V_j}{\sum_j W_{ij} V_j \rho_j^{-1}} \cdot \frac{V_A}{V_I}, \quad (3.41)$$

where

$$V_A = \sum_i \left(\sum_j W_{ij} V_j \right) V_i,$$

and

$$V_I = \frac{1}{\rho_0} \sum_i \rho_i V_i.$$

The reinitialization is a combination of a local operation which acts to ‘blur’ density fluctuations with a distance of $2h$ and a non local part which acts to keep the total mass in the system constant.

3.3 Surface Tension & Contact Angles

Whenever two immiscible fluids meet, the individual molecules which make up both phases feel an imbalance of intermolecular forces. Because the intermolecular forces holding a liquid like water together are attractive this means any interface causes an energy penalty in the system. Thus systems will try and minimize the area of the interface between fluid phases. This energy penalty manifests itself as a force with the form

$$\mathbf{f}_{s.t.} = -\alpha \kappa \hat{\mathbf{n}}, \quad (3.42)$$

where α is the surface tension between the two phases, κ is the curvature of the interface and $\hat{\mathbf{n}}$ is the outward normal of the fluid interface. Most authors use σ or γ to denote surface tension but since σ and γ are already used as part of the SPH formulation we use α . It is possible to show that the curvature corresponds to the divergence of the normal vector giving

a computationally more convenient form for [equation \(3.42\)](#);

$$\mathbf{f}_{\text{s.t.}} = -\alpha \hat{\mathbf{n}} \nabla \cdot \hat{\mathbf{n}}. \quad (3.43)$$

Where a fluid meets a solid boundary there is also an imbalance between the intermolecular forces, however the imbalance is partially offset by attractive interactions between the solid substrate and the fluid. This means that a fluid droplet will also want to minimize the contact between itself and a surface. It will minimize this contact area as much as possible until minimizing it further would cause a corresponding increase in the fluid-fluid surface energy. This balance of surface energies, depicted in [figure 3.5](#), leads to a thermodynamic equilibrium when the fluid and solid interfaces form an angle θ , called the (equilibrium) contact angle. When referring to the contact angle we shall, unless otherwise stated, mean the angle between the solid (wall) and fluid-fluid interface measured relative to the partially wetting phase. Young's equation describes the balance between forces at the contact point due to the differing surface tension, it is

$$\alpha_{ls} + \alpha_{lg} \cos \theta_{ls} = \alpha_{sg}. \quad (3.44)$$

Specifying α_{lg} is important since it sets the speed of any capillary waves generated in the system. Whereas α_{sl} and α_{sg} are less important and are used only to specify the contact angle⁶. For water in contact with air at 20°C $\alpha_{\text{air-water}} = 0.073 \text{ N m}^{-1}$. This decreases with increasing temperature; at 50°C it is reduced to $\alpha_{\text{air-water}} = 0.051 \text{ N m}^{-1}$. Further, the addition of any surfactants also decreases the surface tension. Therefore all simulations in this thesis use a value of $\alpha = 0.05 \text{ N m}^{-1}$, consistent with the the temperature of an operating heap of 50-60°C. Care must be taken that none of the surface tension coefficients are negative. There are three possible cases to consider;

$\theta_{ls} < \pi/2$ The cosine term is positive meaning we can choose $\alpha_{ls} = 0$ which gives $\alpha_{sg} = \alpha_{lg} \cos \theta_{ls}$.

$\theta_{ls} = \pi/2$ The cosine term disappears from the force balance leaving $\alpha_{sg} = \alpha_{sl}$, the choice of this value is then in some sense arbitrary. But to avoid extra time-step constraints and spurious velocities a sensible choice is $\alpha_{sg/sl} = 0$.

$\theta_{ls} > \pi/2$ The cosine term becomes negative but we require $\alpha_{sg} \geq 0 \Rightarrow \alpha_{ls} \geq \alpha_{lg} \cos \theta_{ls}$. Choose $\alpha_{sg} = 0$ giving $\alpha_{ls} = \alpha_{lg} \cos \theta_{ls}$

Thus the surface tension coefficient matrices look like

$$\mathbf{A}_{\theta < \pi/2} = \begin{pmatrix} 0 & \alpha \cos \theta & \alpha \\ \alpha \cos \theta & 0 & 0 \\ \alpha & 0 & 0 \end{pmatrix}, \quad (3.45)$$

$$\mathbf{A}_{\theta = \pi/2} = \begin{pmatrix} 0 & 0 & \alpha \\ 0 & 0 & 0 \\ \alpha & 0 & 0 \end{pmatrix}, \quad (3.46)$$

⁶This is because there can be no capillary waves at a solid interface. For systems with more than two fluids the values will be important.

and

$$\mathbf{A}_{\theta > \pi/2} = \begin{pmatrix} 0 & 0 & \alpha \\ 0 & 0 & -\alpha \cos \theta \\ \alpha & -\alpha \cos \theta & 0 \end{pmatrix}. \quad (3.47)$$

where $\alpha = \alpha_{ls}$, and the rows (columns) are in the order g, s, l .

In multiphase SPH simulations each SPH particle is assigned a colour, which remains constant throughout the simulation. As the SPH particles are advected they carry with them this colour and thus the fluid interfaces are automatically advected. Using the colour we can calculate the normals, and hence curvature of any interfaces. The colour function, defined as

$$C_i^k = \begin{cases} 1 & \text{if } i \in \text{phase } k \\ 0 & \text{otherwise} \end{cases}, \quad (3.48)$$

indicates whether a particle i belongs to phase k . The gradient of C^k at i is calculated using the SPH gradient [equation \(3.18\)](#);

$$\langle \nabla C^k \rangle_i = \frac{1}{\sigma_i} \sum_j \left(\frac{C_i^k}{\sigma_i^2} + \frac{C_j^k}{\sigma_j^2} \right) \nabla W_{ij}. \quad (3.49)$$

This form is the form used throughout this thesis but one can also weight the colour gradient so that most of the colour gradient lies on the heavier phase as in [1]. We have found that while the weighted form from [1] does lead to better interface behaviour it also causes incorrect contact angles with wall boundaries. This issue is not addressed in [1] since they do not consider triple points. The gradient of the colour at an SPH particle points from the point into the other phase, perpendicularly to the interface. This means that the normal can be calculated as

$$\hat{\mathbf{n}}_i^k = \frac{\nabla C_i^k}{|\nabla C_i^k|}.$$

The force given by [equation \(3.43\)](#) acts exactly at the interface, but in SPH, although the particles advect a colour, we do not have a well defined fluid interface on which we can apply the force. For this reason surface tension formulations in SPH usually spread this force over a finite but small region around the fluid interface thereby turning it into a body force rather than a surface effect. To ensure the force is still of the correct magnitude, the function which spreads the force should integrate to unity. Since ∇C is in effect a convolution of the smoothing kernel W with the interface it necessarily integrates to unity (approximately) and so lends itself naturally to this task and the surface tension volume force is

$$\mathbf{f}_{s.t.} = -\alpha |\nabla C| \hat{\mathbf{n}} \nabla \cdot \hat{\mathbf{n}}.$$

Where the divergence of the normal can be calculated using the SPH divergence of $\nabla C/|\nabla C|$. This form for the surface tension force has been successfully applied to simulations where interfaces between only fluid phases exist [1, 56] but it cannot handle contact points or lines, i.e. where fluid-fluid interfaces meet a solid boundary, without further modifications.

A widely used scheme which is capable of handling contact angles and multiphase surface energies is the *Continuum Surface Force* (CSF) model of Brackbill et al. [18, 49], in which the various colours' gradients are combined into interfacial stresses and the force is then calculated as a divergence of this stress. The stress at a particle i (belonging to phase k) due to phase l is given by

$$\boldsymbol{\Pi}_i^{kl} = \frac{\alpha_{kl}}{|\nabla C_i^l|} \left(\frac{|\nabla C_i^l|^2}{d} \mathbf{I} - \nabla C_i^l \otimes \nabla C_i^l \right). \quad (3.50)$$

These stresses are summed to give the total stress

$$\boldsymbol{\Pi}_i = \sum_{l \neq k} \boldsymbol{\Pi}_i^{kl}, \quad (3.51)$$

the divergence of which gives the force

$$\mathbf{f}_{\text{s.t.}} = \sigma_i \sum_j \left(\frac{\boldsymbol{\Pi}_i}{\sigma_i^2} + \frac{\boldsymbol{\Pi}_j}{\sigma_j^2} \right) \cdot \nabla W_{ij}. \quad (3.52)$$

The CSF model has been successfully used for a number of studies but it is not the only option for simulating surface tension forces in SPH. A different method, that works directly with the colour gradients, is introduced and discussed in [section 5.7.4](#).

3.4 Timestepping

The previous sections detailed how the gradients, etc, that occur in the Navier-Stokes equation are discretized in space and appropriately approximated. Once the accelerations on each particle have been calculated, the velocities and positions must be integrated with time to move the simulation forward. For this reason a suitable integration scheme (timestepping scheme) must then be applied to the equations of motion.

A variety of timestepping schemes have been used in the SPH literature, many studies use the leap-frogging scheme [60]. In this thesis a predictor corrector scheme is used. The position of all the particles at a time t is updated to give the new positions at a time $t + \Delta t$ later. For each particle i its position can be updated using the following set of equations;

Halfstep
$\mathbf{x}_i^{(t+\Delta t/2)} = \mathbf{x}_i^{(t)} + \frac{\Delta t}{2} \mathbf{u}_i^{(t)} \quad \mathbf{u}_i^{(t+\Delta t/2)} = \mathbf{u}_i^{(t)} + \frac{\Delta t}{2} \mathbf{a}_i \left(\{\mathbf{x}_j^{(t)}\}, \{\mathbf{u}_j^{(t)}\} \right)$
Fullstep
$\tilde{\mathbf{x}}_i^{(t+\Delta t/2)} = \mathbf{x}_i^{(t)} + \frac{\Delta t}{2} \mathbf{u}_i^{(t+\Delta t/2)} \quad \tilde{\mathbf{u}}_i^{(t+\Delta t/2)} = \mathbf{u}_i^{(t)} + \frac{\Delta t}{2} \mathbf{a}_i \left(\{\mathbf{x}_j^{(t+\Delta t/2)}\}, \{\mathbf{u}_j^{(t+\Delta t/2)}\} \right)$ $\mathbf{x}_i^{(t+\Delta t)} = 2 \tilde{\mathbf{x}}_i^{(t+\Delta t/2)} - \mathbf{x}_i^{(t)} \quad \mathbf{u}_i^{(t+\Delta t)} = 2 \tilde{\mathbf{u}}_i^{(t+\Delta t/2)} - \mathbf{u}_i^{(t)}$

The notation $\{\cdot\}$ is used to indicate that the accelerations are functions of all particles' positions

and velocities. Also if one is updating the density by integrating [equation \(3.21\)](#) rather than using direct density then the density at one time step is related to the density at the next by

Halfstep
$\rho_i^{(t+\Delta t/2)} = \rho_i^{(t)} + \frac{\Delta t}{2} \frac{d\rho_i}{dt} \left(\{\mathbf{x}_j^{(t)}\}, \{\mathbf{u}_j^{(t)}\} \right)$
Fullstep
$\tilde{\rho}_i^{(t+\Delta t/2)} = \rho_i^{(t)} + \frac{\Delta t}{2} \frac{d\rho_i}{dt} \left(\{\mathbf{x}_j^{(t+\Delta t/2)}\}, \{\mathbf{u}_j^{(t+\Delta t/2)}\} \right)$ $\rho_i^{(t+\Delta t)} = 2\tilde{\rho}_i^{(t+\Delta t/2)} - \rho_i^{(t)}$

3.4.1 Timestep Criteria

The time stepping scheme detailed above increments the particles' positions in finite time steps of size Δt . The choice of Δt is important; because of the explicit nature of the timestepping scheme Δt must be smaller than certain values otherwise the simulation will become unstable. Conversely, the shorter Δt is the longer the simulations take. There are several timestep criteria, all of which derive from the CFL condition. The CFL condition can be thought of as ensuring that no wave in the system can move far enough in a single Δt that we lose track of its phase. The first condition is the velocity CFL condition

$$\Delta t_{CFL} = \min_{i,j} \left\{ \frac{h}{|\mathbf{u}_{ij}|} \right\}. \quad (3.53)$$

The acceleration timestep condition prevents particles gaining too much velocity in a single timestep;

$$\Delta t_F = \min_i \left\{ \sqrt{\frac{h}{|\mathbf{a}_i|}} \right\}. \quad (3.54)$$

This is important because if a particle moves too far in a single timestep it can overshoot the overlap region between processes meaning it will not be transferred correctly (see [section 4.1.6](#)). Additionally, since the timestep criteria is calculated at the predictor step, this condition prevents particles stepping onto each other. This is important to avoid since it would lead to a division by zero in [equation \(3.32\)](#). The acoustic timestep condition

$$\Delta t_A = \min_i \left\{ \frac{h}{c_0 \sqrt{(\rho_i^*)^{\gamma-1}}} \right\}, \quad (3.55)$$

ensures stability for sound waves in the system. There is also the capillary wave timestep limit which, unlike the other timesteps, does not depend on the individual particles' properties and

is not updated as the simulation progresses. It is given by

$$\Delta t_{CW} = \sqrt{\frac{h^3}{2\pi} \min_{k,l} \left\{ \frac{\rho_0^k + \rho_0^l}{\alpha_{kl}} \right\}} \quad (3.56)$$

where the minimum is taken over the different pairs of phases, denoted by the indices k and l . This timestep comes from considering the capillary wave phase velocity

$$v_p^{kl} = \sqrt{\frac{\alpha_{kl}|\mathbf{k}|}{\rho_0^k + \rho_0^l}} \quad (3.57)$$

with $\mathbf{k}$ the wavenumber vector. The maximum resolvable wavenumber equates to a wavelength of $\sim h$ thus $k_{\max} = 2\pi/h$, substituting this into the phase velocity and then this into the CFL conditon gives [equation \(3.56\)](#).

Explicit schemes for diffusion equations are also subject to a timestep limit of the form $\Delta t < \Delta x^2/2D$ where D is the diffusion coefficient. Since the kinematic viscosity can be thought of as quantifying the ‘diffusion of momentum’ in the system this introduces yet another limit on Δt ; the viscous diffusion timestep

$$\Delta t_{VD} = \min_k \left\{ \frac{h^2 \rho_0^k}{2\mu^k} \right\}. \quad (3.58)$$

In most situations this timestep is not the limiting factor but for very small systems it can become very constraining since it scales as h^2 .

For the simulation to be stable we require

$$\Delta t < \min\{\Delta t_{CFL}, \Delta t_A, \Delta t_F, \Delta t_{CW}, \Delta t_{VD}\}$$

so we choose

$$\Delta t = C \min\{\Delta t_{CFL}, \Delta t_A, \Delta t_F, \Delta t_{CW}, \Delta t_{VD}\} \quad (3.59)$$

with $C < 1$. Typically the convergence factor C is given the value 0.2. [Figure 3.6](#) shows the relationship between smoothing length and the timestep for the most important timestep conditions for water with a numerical speed of sound of $c_0 = 10 \text{ ms}^{-1}$. For larger scale simulations ($h \gtrsim 10^{-5} \text{ m}$) the most limiting constraint is the acoustic timestep limit, at intermediate scales ($10^{-8} \text{ m} \lesssim h \lesssim 10^{-5} \text{ m}$) the capillary constraint becomes the most constraining then at scales below this ($h \lesssim 10^{-8} \text{ m}$) the viscous diffusion becomes the limiting constraint. In this thesis the multiphase simulations have a speed of sound typically slightly less than 10 ms^{-1} which means that in most cases the capillary constraint is dominant.

3.5 Pressure Matching

In the absence of any shock fronts the pressure throughout a physical system is a continuous function. This has implications for the equations of state used in multiphase SPH simulations.

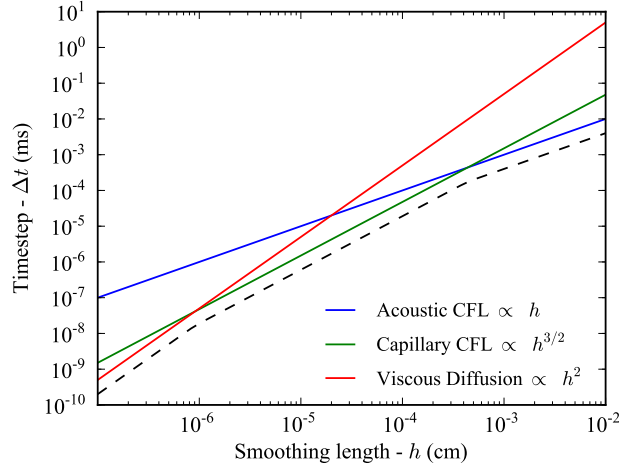


Figure 3.6: The timestep Δt plotted against smoothing length h for water with numerical speed of sound $c_0 = 10\text{m s}^{-1}$. The three solid lines show the usually limiting factors and the dashed black line represents the timestep as given by [equation \(3.59\)](#).

For example consider a flat interface between air and water; at equilibrium the pressure on both sides of the interface must be equal otherwise the interface SPH particles would be accelerated. Since the relative density, $\rho^r = \rho/\rho_0$, on both sides of the interface is equal, we need both equations of state (for the air and for the water) to be equal. This ensures that the resulting pressure does not have a discontinuity, i.e. $P_l = P_g$. Since we are using the Tait equation the bulk modulus depends on the fluid density and the (numerical) speed of sound and we have

$$\frac{\rho_{0,l} c_{0,l}^2}{\gamma} \left((\rho_l^r)^\gamma - 1 \right) = \frac{\rho_{0,g} c_{0,g}^2}{\gamma} \left((\rho_g^r)^\gamma - 1 \right),$$

using the fact that $\rho_g^r = \rho_l^r$ we can cancel and rearrange:

$$c_{0,g} = c_{0,l} \sqrt{\frac{\rho_{0,l}}{\rho_{0,g}}}. \quad (3.60)$$

Choosing the speed of the sound in the g phase in this way ensures that the Tait equations of state for both phases are equal for equal relative densities and that we do not see spurious accelerations at the interfaces. However, it does mean that the speed of sound is higher in the *lighter* phase and as a consequence the compressibility is also lower. For an air water system the speed of sound in the air phase is therefore a factor of $\sqrt{1000\text{kg}/1.1\text{kg}} \approx 30$ larger than in the water phase. Additionally since the acoustic timestep is inversely proportional to c_0 this increases the number of iterations required to simulate the same amount of real time by a factor of 30.

3.6 Smoothing Kernels

Up until now we have not been specific about the form of the smoothing kernel beyond the constraints it must satisfy. Given the SPH formalism defined in [section 3.1](#) one is in principle

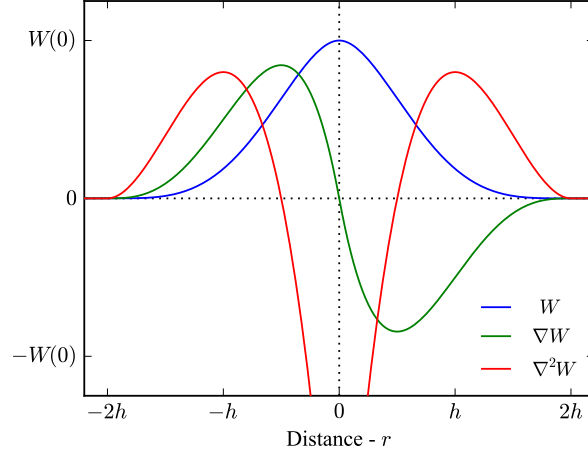


Figure 3.7: The quintic spline kernel in 1D plus its first and second derivatives, shown in blue, green and red respectively.

free to choose any function which satisfies the constraints in [table 3.1](#), however in practice the choice of kernel can have a big impact on the performance of any given SPH simulation. For example a Gaussian kernel satisfies the above constraints and has the desirable feature that it is infinitely differentiable everywhere (smooth). There are however two practical drawbacks to using a Gaussian kernel; it involves computing expensive exponential functions and it is non-zero everywhere. It is thus not a good choice from a computational viewpoint.

More usual is the use of kernels with only finite support thus eliminating the need to sum over all the particles in the system, as well as kernels which are easy to compute in as few operations as possible.

The cubic spline kernel was first defined in [\[77\]](#). It is simple to compute, but, being cubic, its Laplacian is piecewise linear. This has severe implications for the accuracy of the SPH approximation of, for example, viscosity. To counter this many authors use higher-order kernels, sometimes specifically designed for the viscosity calculation. The quartic spline kernel was introduced by Liu et al. in [\[68\]](#), it has a differentiable Laplacian rendering it more suitable for viscosity calculations. Müller et al. [\[80\]](#) introduced what they call the Poly-6 kernel which has the advantage that it can be computed without an expensive *if*-statement rendering it especially useful on GPU hardware where branching is a costly bottleneck.

Our SPH code has the option to use a variety of smoothing kernels including cubic, quartic and quintic splines, but generally the Wendland quintic spline is used. It was defined by Wendland in [\[101\]](#) as

$$W(r, h) = C \begin{cases} \left(1 - \frac{r}{2h}\right)^4 \left(1 + \frac{2r}{h}\right) & \text{if } 0 \leq \frac{r}{h} < 2 \\ 0 & \text{otherwise} \end{cases}, \quad (3.61)$$

with derivative

$$\left. \frac{dW(r', h)}{dr'} \right|_{r'=r} = C \begin{cases} \frac{5}{8h} \frac{r}{h} \left(2 - \frac{r}{h}\right)^3 & \text{if } 0 \leq \frac{r}{h} < 2 \\ 0 & \text{otherwise} \end{cases}. \quad (3.62)$$

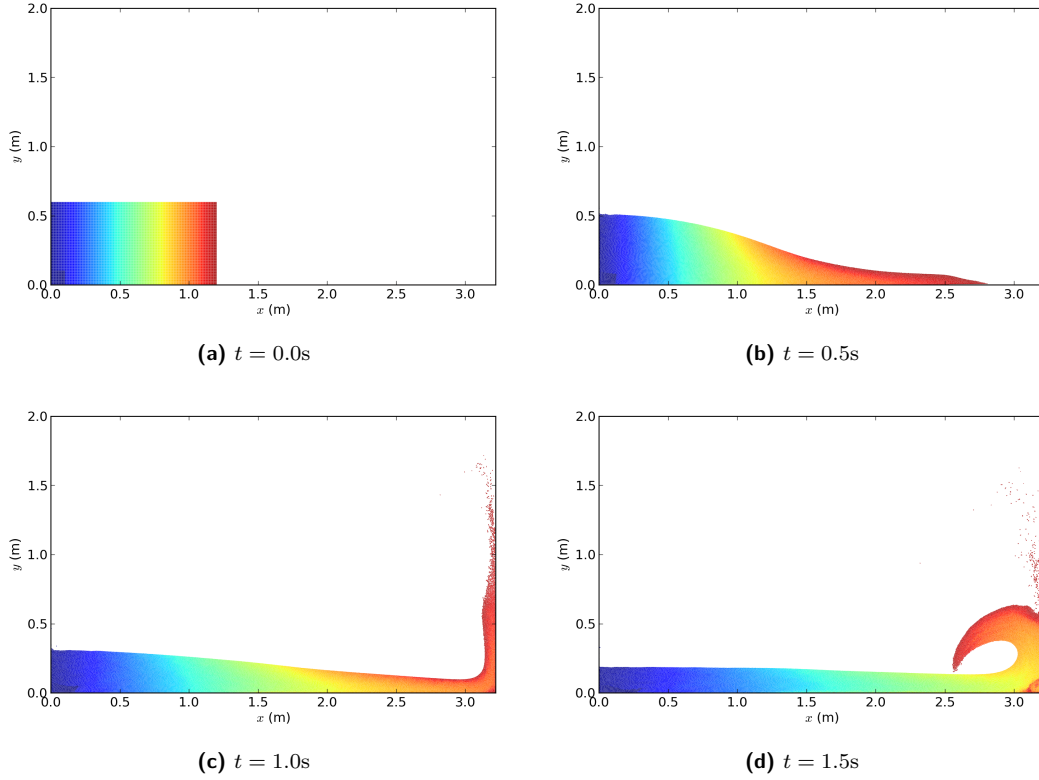


Figure 3.8: Snapshots from an example SPH simulation of a ‘dam break’; where an initially stationary rectangle of fluid is released inside a container that is 3m by 2m. The particles are coloured according to their initial location within the rectangle

C is a normalization constant which is determined by the dimension one is working in:

- 1D $C = \frac{3}{4h}$
- 2D $C = \frac{7}{4\pi h^2}$
- 3D $C = \frac{21}{16\pi h^3}$

The form of this kernel is shown for 1D in figure 3.7. It is not piecewise on $[0, 2h)$ meaning it can be computed quickly and, because it is smooth⁷, it also leads to accurate estimates. The Wendland kernel is becoming more common as the kernel of choice for SPH simulations displacing the previously ubiquitous cubic-spline [3, 37, 38, 76].

The equations detailed in this section are sufficient to produce a working SPH simulation. Shown in figure 3.8 are four snapshots, at $t = 0.0, 0.5, 1.0$ and 1.5 s, of a ‘dam break’ simulation which is a canonical test for SPH codes [8, 24, 27, 61]. The particular geometry used here is taken from Colagrossi [24] and the simulation was produced using our SPH code, the computational aspects of which are detailed in the next chapter.

⁷Except at $r = 0$ where W''' is discontinuous.

3.7 Conclusions

This chapter detailed the foundations of the SPH method including the mathematical framework and how one can derive the discretised equations of motion. Within this framework we have implemented surface tension and contact angle forces using the CSF method. The CSF model has been well established as a method capable of capturing the necessary behaviour of immiscible fluid mixtures and lends itself naturally to SPH. These factors, combined with SPHs innate ability to handle the advection of interfaces as well as topological changes (e.g. droplet formation or coalescence), make SPH a natural choice for simulating the flows seen in packed beds.

The chosen combination of corrected kernel (equation (3.16)) and quintic smoothing kernel (equation (3.61)) was confirmed to give the expected second order converge for smoothing factors > 1.3 for a test case of 2D flow in a pipe. The gradient estimator equation (3.19) allows us to calculate the gradients accurately even in the presence of particle disorder, for example in the test case shown in figure 3.8. This will be revisited in chapter 5 where a new boundary formulation is incorporated with this particle inconsistency correction. Additionally, the handling of fluid interfaces with large density ratios was discussed the solution of matching speeds of sounds to ensure a continuous pressure throughout the domain was detailed.

This chapter has focused on the mathematics behind the SPH method and not on the actual computation or key aspects such as including wall boundaries, these are issues which are discussed in chapters 4 and 5 respectively.

Chapter 4

Implementation & Computation

The previous chapter detailed the mathematical underpinnings of the SPH method applied to simulating the Navier-Stokes equation. In order to perform useful simulations it is necessary to translate these ideas into a functioning numerical code. Several computational issues must be addressed to achieve this; these are discussed in this chapter. Recent studies have looked at using GPU technology to accelerate SPH codes, both on a single GPU [61] and split across several GPUs [42]. Though more recent models have alleviated these issues, at the time this work was begun GPUs suffered from poor performance when using double precision data and a single GPU would not have had enough memory for the larger simulations with $>10^6$ particles. Thus the simulations in this thesis were all carried out in a distributed computing environment on tens to hundreds of cores.

4.1 The SPH Code

The SPH numerical code used for this thesis was developed ‘in house’ allowing for complete customisation and control. The simulation code as well as pre- and postprocessing codes were written in C++, utilizing the Message Passing Interface (MPI) library to efficiently parallelise the code. The postprocessing software outputs data in VTK format making it amenable to visualization using the Paraview¹ software. The SPH simulation code was profiled using the ‘gprof’ utility which enabled us to improve the speed drastically and remove bottlenecks in the code. This is an important step for all high performance numerical codes, but it is especially important for SPH because it requires, for a given resolution, more computational power than competing methods. This is due to the fact the simulation resolution is given by the smoothing length h but the domain is discretized at the length scale $\Delta x < h$.

4.1.1 Asymptotic Complexity

The first papers to use SPH used a Gaussian distribution for their smoothing kernel [40, 72]. Because the Gaussian distribution has support everywhere all of the sums in the SPH estimators (equation (3.31) for example) run over every particle in the domain. Since the forces need to be calculated for all N particles in the domain and each force calculation has to go over all N particles, this leads to an asymptotic complexity of $O(N^2)$, i.e. doubling the number of particles increases the required runtime by a factor of four.

This poor scaling would severely hamper the utility of SPH for practical simulations. For this reason the constraint of finite kernel support is introduced (see section 3.1). The finite support

¹<http://paraview.org/>

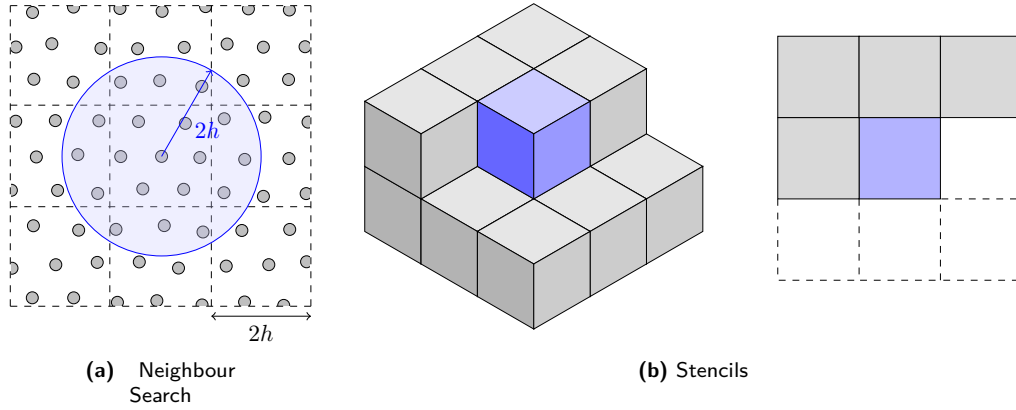


Figure 4.1: Illustration of the linked cell grid. Finding particles within $2h$ only requires searching nearby grid cells. To avoid double counting particles only certain neighbouring cells are checked, the stencils for which are shown for both 2D and 3D.

radius $2h$ means that each particle can only see a subset of the total particles. On average a particle has

$$N_{\text{neighbour}} = \lfloor 4\pi s^2 \rfloor - 1,$$

and

$$N_{\text{neighbour}} = \left\lfloor \frac{32}{3}\pi s^3 \right\rfloor - 1,$$

neighbours in 2D and 3D respectively, where $s = h/\Delta x$ is the smoothing factor.

The fact that the number of neighbours is (approximately) the same for every particle in the domain means that now our theoretical complexity is reduced to $O(N \cdot N_{\text{neighbour}})$ and since $N_{\text{neighbour}}$ does not scale with the system size or N in any way, the asymptotic complexity is $O(N)$. This is as good as a numerical method can get and, all things being equal, means that if we double the system size we require only twice as much computational power to simulate it.

4.1.2 Linked Cell Grid

In practice, to achieve the theoretical complexity of $O(N)$ one must know which other particles are within $2h$ of each particle. For each particle i one cannot simply check all other particles j as this would return us to a complexity of $O(N^2)$. To reach the desired complexity our SPH code utilises a method known as the *linked cell grid* method.

The computational domain Ω is decomposed into small cubic (square) cells of side $2h$ [41]. To search for particles within a radius of $2h$ we therefore only need to search our cell and neighbouring cells. Each cell has a 2D/3D ‘subscript’ index which can be directly calculated from a position. This means that given a particle i we can calculate in which cell it belongs, by using its position. This ‘subscript’ can then be converted into a 1-dimensional index by the function $\text{idx}(\mathbf{r})$. Each particle is then stored in the corresponding cell covering its location. This means that when searching for particles j near a particular particle i we need only work

out which cell i is in then we can check the particles residing in the neighbouring cells of $\text{idx}(\mathbf{r}_i)$ and the cell $\text{idx}(\mathbf{r}_i)$ itself.

The number of cells in each dimension is given by

$$N_\alpha = \left\lceil \frac{L_\alpha}{2h} \right\rceil,$$

where L_α is the length of the domain in the α -dimension. These are then used to get the one dimensional index from a position

$$\text{idx}(\mathbf{r}) = \left\lfloor \frac{r_z}{2h} \right\rfloor + \left\lfloor \frac{r_y}{2h} \right\rfloor \cdot N_z + \left\lfloor \frac{r_x}{2h} \right\rfloor \cdot N_y N_z. \quad (4.1)$$

In two dimensions this becomes

$$\text{idx}(\mathbf{r}) = \left\lfloor \frac{r_y}{2h} \right\rfloor + \left\lfloor \frac{r_x}{2h} \right\rfloor \cdot N_y,$$

because $N_z = 1$ and $r_z = 0$.

Some of the particles inside neighbouring cells will be outside of the $2h$ radius but we must still search them to check. This means that, for a given particle, to calculate any SPH sums we need to loop over approximately

$$3^d \frac{(2h)^d}{\Delta x^d} = (6s)^d$$

SPH particles. Taking $s = 1.3$ this gives ≈ 60 particles in 2D and ≈ 470 particles in 3D. Given that there will also be more particles in the system, 3D simulations represent a challenge computationally.

Stencils

A naive implementation of the SPH sums would check each pair of particles twice. Consider neighbouring particles a and b , while checking a 's neighbours we see b and while checking b 's neighbours we see a . By doing this, the code is performing unnecessary searches through the lists of SPH particles, a potentially slow operation.

To avoid this, rather than checking all neighbouring cells of a particle, our code uses a stencil to check only certain neighbouring cells (see [figure 4.1b](#)). The shape of the stencils in 2D and 3D means that we will still hit all possible pairs of nearby particles but will only see each pair once. Now, because we only see each pair once, we update not just the relevant values for a but for b as well.

Care must be taken with the middle cell (highlighted blue in [figure 4.1b](#)) to avoid hitting pairs twice. When calculating the quantities for a particular particle, a say, in the central cell we only add contributions from particles stored after a in the data structure. This is because particles stored before a will have added their contribution to a already. In this way we can update both particles at once and avoid double counting.

The use of stencils gives slightly less than a twofold speed up when calculating SPH sums

because we cut in half the number of loops over neighbouring particles. This is a significant speed-up which should not be ignored when implementing such particle-particle codes.

4.1.3 Boundary Particles

All simulations carried out for this thesis use a single layer of stationary particles at the wall boundaries. This is not inherently necessary for the SPH method but it is useful from both a numerical and computational perspective. Many of the necessary computations for handling wall boundaries, a detailed discussion of which is deferred to [chapter 5](#), require knowing information about the domain boundary in the vicinity of an SPH particle. Therefore it is necessary to be able to quickly find those boundaries which are within $2h$ of a given particle. By storing boundary particles in the linked cell grid discussed above, one can utilise its speed to aid searching for boundaries. Because of this the first step taken by the SPH after loading the configuration files is to place a layer of boundary particles with average spacing Δx along any wall boundaries.

The boundary particles interact with the SPH particles as if they were normal particles but they are fixed for the duration of the simulation. While it is not necessary to have the boundary particles interact as usual (they could be used merely to provide a lookup for boundary objects) it is found that doing so improves containment of SPH fluid particles within the domain.

4.1.4 Periodicity

Because we are interested in simulating the interior of heaps we do not want any boundary effects. To achieve this we take representative elementary volumes (REVs) which are periodic in all directions thereby avoiding wall effects, which are known to have large effects on both saturated and unsaturated flows. The periodic nature of the boundaries means displacements must be calculated accounting for the periodic distance, rather than the naïve Euclidean distance. The periodic domain Ω is a cuboid shaped subset of $\mathbb{R}^d$ given by

$$\Omega = \prod_{\alpha=1}^d [0, L_{\alpha}),$$

where L_{α} is the length of the period in the α Cartesian direction. So, in two dimensions we have $\Omega = [0, L_x) \times [0, L_y)$. The periodic displacement function $\mathbf{d}\mathbf{x}\mathbf{p}$ is a vector, whose α -component is given by

$$(\mathbf{d}\mathbf{x}\mathbf{p}(\mathbf{x}, \mathbf{y}))^{\alpha} = \begin{cases} x^{\alpha} - y^{\alpha} & \text{if } |x^{\alpha} - y^{\alpha}| \leq L_{\alpha}/2 \\ x^{\alpha} - y^{\alpha} - L_{\alpha} & \text{if } (x^{\alpha} - y^{\alpha}) > L_{\alpha}/2 \\ x^{\alpha} - y^{\alpha} + L_{\alpha} & \text{if } (x^{\alpha} - y^{\alpha}) < -L_{\alpha}/2 \end{cases} \quad (4.2)$$

It is the shortest vector between two points in the domain accounting for the periodicity. $\mathbf{d}\mathbf{x}\mathbf{p} : \Omega \times \Omega \rightarrow \mathcal{D}_p$, where $\mathcal{D}_p = \prod_{\alpha=1}^d [-L_{\alpha}/2, L_{\alpha}/2]$. The magnitude $|\mathbf{d}\mathbf{x}\mathbf{p}(\mathbf{x}, \mathbf{y})|$ forms a valid metric for the periodic domains used.

In our code, particle to particle interactions are calculated using the normal Euclidean distance. Periodicity is accounted for by creating a virtual copy of every particle within $2h$ of a period. This virtual particle has its position updated to be that of a periodic image of the original particle. In this way periodicity is handled with no change to the core routines of the SPH code. Additionally, for parallel simulations the virtual particles can be transferred onto their respective processes to handle domain padding and parallelization, see [section 4.1.6](#). The linked cell grid for each process is extended by two cells ($4h$) in each direction to handle the ‘virtual’ particles, be they periodic or from neighbouring processes.

Solid boundaries can and do cross the domain periods and so must also be handled carefully. Any calculations involving the boundary, such as the distance of an SPH particle to the wall, must use [equation \(4.2\)](#). Computationally the reason for this is that interactions are calculated relative to the boundary object itself rather than creating virtual periodic images.

4.1.5 Data Structures & Code Optimization

Bad memory locality has a severe deleterious effect on code performance. If are reading a quantity from memory the resulting read time will be significantly larger if the CPU has to fetch the data from RAM rather than either the L1 or L2 cache. For a modern Intel i7 or Xeon processor reading memory from L1 cache takes approximately 5 clock cycles, where as reading from RAM can take several thousands of cycles (~ 100 ns)².

Internally to our simulation code the SPH particles are stored in a doubly linked ‘mutlilist’ data structure; one linked list for storing the particles and another for placing them into the linked cell grid. Programmatically this is achieved *intrusively*, meaning the necessary pointers for forming the list structure are stored inside the SPH particle objects themselves. This intrusive design increases memory locality because the particles’ data are stored next to the pointers for the linked list in memory.

Further, the SPH particles are initially allocated in large blocks, which means that they are in contiguous memory locations. When particles are no longer needed, for instance because they moved off the processor’s sub-domain, they are added to a store. When a new particle is needed it is taken from this store. Using a store like this both avoids the need to allocate and deallocate memory repeatedly while the simulation is running (a generally slow operation) and improves memory locality by ensuring we are always using memory from the originally allocated block.

Modern CPUs include facilities for so-called prefetching, whereby data is loaded from memory at the same time as the CPU is performing calculations. A modern optimizing compiler will utilize this when it can, but, as with all optimizations, the compiler cannot always tell when it can safely do so. One can manually call the prefetching routine using compiler intrinsics. Intrinsics are a way of directly calling CPU instructions from a higher level language than assembly. We attempted to improve the performance of our SPH code by utilising this manual

²https://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf
- retrieved 31 August 2014.

prefetching but found that it made no discernible difference to the speed of the calculations. The speed of the code was measured by the reported CPU clock and wall times as well as using a profiler. Because of this, prefetching was not included in the code in the long term.

The most probable reason that prefetching makes no difference is due to the linked list structure used. Since the pointers are stored intrusively, in order to look two items ahead in the list we first need to go to the location of the particle one ahead in order to tell the compiler from where in memory it should prefetch. The location of the particle one ahead in the list is not necessarily adjacent to the current particle in memory. This will always be an issue for looping over neighbouring particles (since they can be stored in any order depending on the fluid's state). For performing calculations which do not depend on neighbouring particles (e.g. timestepping) this limitation could be avoided by storing them in a contiguous array. This, however, introduces more overhead and complexity for handling the removal or reordering of particles within the data structure.

Our linked-cell-grid index function $\text{idx}(\mathbf{x})$ is, in effect, a raster over the cells of the grid. Therefore cells which neighbour each other in the fastest varying dimension will be far away from each other in memory despite being close in space. For the data structure employed in our code this is not an issue since the particles are not stored in the linked-cell-grid itself. If, however, one were to actually store the particles in the linked-cell-grid type data structure the indexing scheme given by [equation \(4.1\)](#) could lead to poor performance. A better solution in this case might be a Hilbert-like space filling curve but this adds extra complexity to the calculation of $\text{idx}(\mathbf{x})$ and this would have to be weighed against any increase in memory locality.

The code was also extensively profiled to maximise code performance and find bottlenecks. As an example, the `max` function from the C++ standard library is used inside the inner simulation loop to clamp values to a specific range and thereby avoid a costly branching *if*-statement. The repeated calls to `std::max` were found to account for a significant fraction of the simulations' run times. Replacing this with a compiler intrinsics call³ was found to completely eliminate this slowdown. Additional speed-ups were achieved by prompting the compiler to apply optimizations such as inlining certain function calls.

4.1.6 Parallelisation

In order to perform large, high resolution simulations we require a lot of computational power, therefore the SPH code has been written explicitly to be massively parallel. It can be run on any number of processes. In this thesis simulations have been run on between 10 and 720 cores depending on the number of SPH particles required. As with many parallel simulation codes, the method of parallelisation which lends itself most readily to SPH is *domain decomposition*, whereby each processor is responsible for a separate part of the spatial domain.

³`_mm_store_sd( &a, _mm_max_sd(_mm_set_sd(a),_mm_set_sd(b)) );`

Domain Decomposition

Domain decomposition works by splitting the simulation domain into smaller subregions then letting each separate processor perform only the calculations for a particular subregion. Because each cell of the linked cell grid needs to be able to see its neighbouring cells, each process must send its outermost layer of cells to neighbouring processes and also receive its neighbours' outermost cells into a padded region. Thus, at each time-step data must be exchanged between neighbouring processes. This data exchange adds an overhead to the simulation which will depend on the number of processes and the shape of the subregions. The optimal shape for a subregion is one which minimizes amount of data which needs transferring. This is equivalent to minimizing the perimeter in 2D and the surface in 3D. Since the domains must be rectangular and cuboidal respectively, the optimal shapes for the subregions are those closest to a square or cube.

A typical measure of the effectiveness of a parallelisation scheme is the 'speed up' S_p . It is the ratio of the time to complete some operation on one processor compared to the time on many processors. It is a function of the number of processors N_p . The ideal scaling for a parallel computation is linear $S_p \sim N_p$. This is very hard to achieve in practice if there is some interdependence between the processes.

For our system the expected scaling of S_p with N_p in 3D can be shown to be

$$S_p(N_p) = \frac{N_p}{A + B \sqrt[3]{N_p}}, \quad (4.3)$$

for some constants A and B , if one takes cubic subregions. Thus $S_p \sim N_p^{\frac{2}{3}}$, though this scaling is only true for an idealized system (i.e. one in which there is always more time spent calculating than communicating) as $N_p \rightarrow \infty$. For a real simulation, if one increases N_p too much then we see performance much worse than $N_p^{\frac{2}{3}}$ because the processors do not have enough computation to do and so the data transfer comes to dominate the time expended by each subdomain.

If one takes instead slices of the domain or pillars instead of approximately cubic regions the resulting scaling of S_p is less favourable, i.e. further from $S_p \sim N_p$. For the derivation of [equation \(4.3\)](#) see [appendix A.1](#). For small N_p the speed up is approximately linear and how quickly S_p deviates from this linear relationship depends on the exact simulation being performed; for a simulation with more SPH particles the scaling would stay closer to $S_p \propto N_p$. This is because the communication represents a relatively smaller cost than for the case where there are fewer SPH particles on the same number of cores.

The relationship given by [equation \(4.3\)](#) shows that as we split the domain into more subdomains, the processes each spend relatively more time communicating with each other as compared to actually computing (for a fixed resolution). This increasing communication demand is seen in the falling efficiency. The efficiency is the ratio of the speed up to the number of processors, $E_p = S_p/N_p$. It is effectively a measure of what proportion of time is actually being spent computing as opposed to sending data or waiting for other processes. As can be seen in [figure 4.2](#) the speed up given by [equation \(4.3\)](#) fits very well with $A = 1$ and $B = 0.4$.

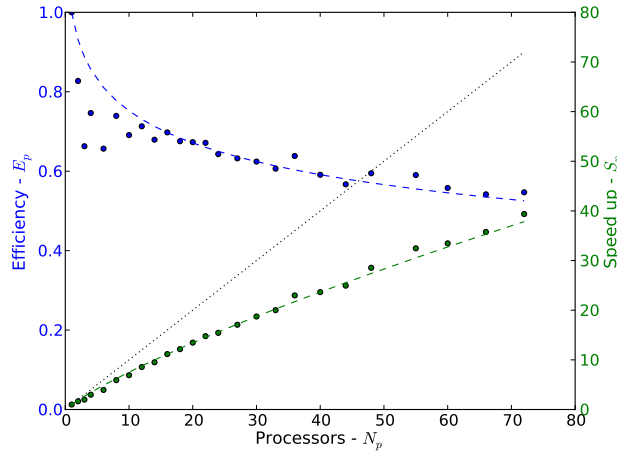


Figure 4.2: Graph showing the speed up and efficiency of the SPH code running a 3D simulation for an increasing number of processors. The dotted black line shows the ideal case of $S_p = N_p$ and the dotted coloured lines show the expected values given by [equation \(4.3\)](#).

Load Balancing

For simulations with free surfaces or complex boundaries there can be more SPH particles in one part of the domain than elsewhere. This means that if we just divide the domain into approximately equal blocks then some processors will be doing more work while others sit idle. This will cause extra inefficiency in the code as we must wait for all processes to finish their computations before we can move on to the next iteration. To counter this the SPH code includes a load balancing routine, which periodically resizes the subdomains to ensure that each processor has roughly the same number of particles.

Flood Filling Algorithm

Because the geometries of our systems are complex, with the interstitial space forming networks of connected channels we must take care when placing the SPH particles into the domain. A general way of filling arbitrary geometries is to use a flood filling algorithm. Flood fill algorithms work by starting with an initial point which is known to be inside the domain, the neighbours of this point can then be checked to see if they are in the domain. If an odd number of walls are intercepting the line between the two points, we know that the new point must lie outside the domain and therefore is invalid. Whenever a valid point is found it can be added and its neighbours can then be checked for validity.

The checking proceeds through the domain on a regular cubic grid placing particles on sites that neighbour valid particles and are themselves valid. Rastering in one particular direction leads to ‘shadows’ behind geometries (because a particle can only be marked valid once one of its neighbours has been), thus at each step the raster direction is swapped. These steps are illustrated in [figure 4.3](#).

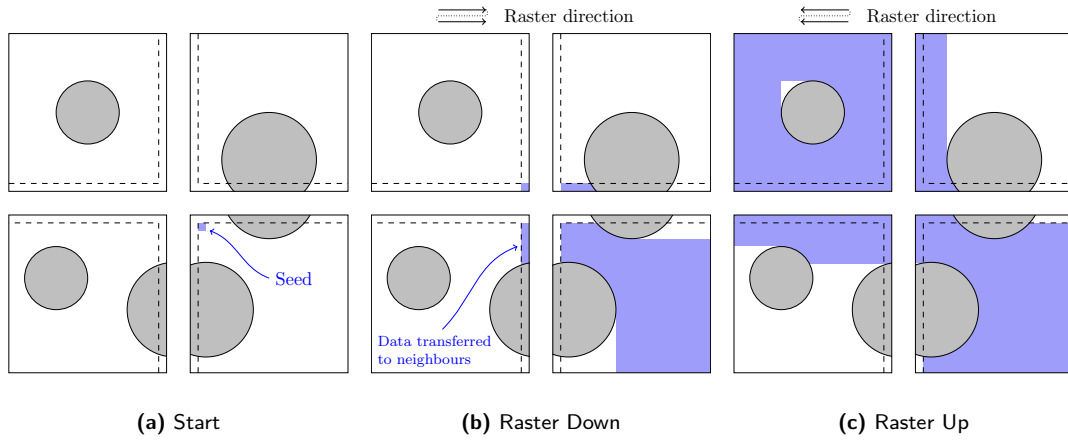


Figure 4.3: An illustration of the flood filling algorithm. The raster steps are performed, and data is transferred, until no new SPH particles can be placed, meaning the domain is fully filled.

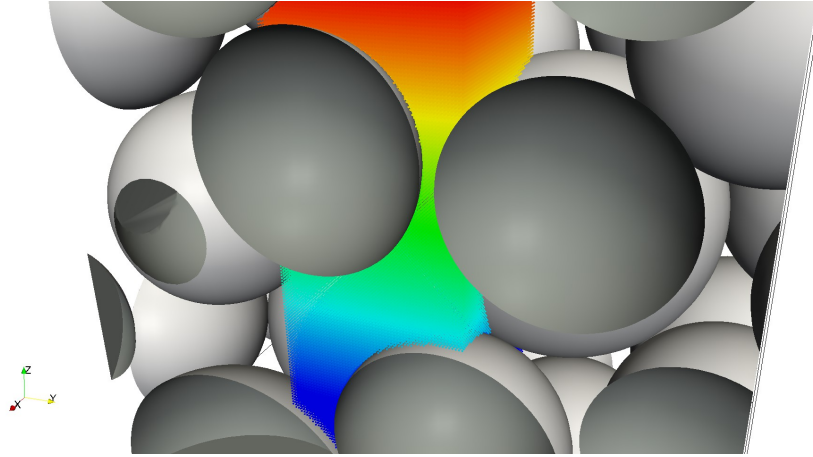


Figure 4.4: An example of a 3D domain which has been flood filled using the parallelized algorithm described. Only the water phase is shown. The colour corresponds to the particles' positions in space.

The basic flood fill algorithm was adapted to maximize the speed. Firstly, when checking if a particle is valid the algorithm must examine the walls in the system to see if they intercept. In systems with possibly thousands of walls this makes the algorithm prohibitively expensive. For this reason before flood filling, the SPH wall particles are placed into the linked-cell-grid and we can use this to quickly search for nearby walls, thereby significantly reducing the number of walls that need to be checked for each SPH fluid particle.

Secondly, the algorithm was fully parallelised in line with the domain decomposition, meaning each processor can raster its own domain separately. After each rastering step the data in the padded regions of the domains is transferred to allow the filling to cross processor boundaries. This means that initially only the processor whose subdomain contains the seed point is placing particles on the grid, but once the rastering reaches a process boundary it is transferred to the neighbour and the neighbour can then start filling its domain while the initial processor

continues filling itself. Because the filling cannot cross the whole domain in one iteration (it can only cross a subdomain) splitting the domains like this actually leads to it taking *more iterations* to fill the domain than if we had not split the domain, but crucially, because the subdomains can be filled in parallel it still requires far less time. Another benefit is that the flood filling algorithm has a large memory overhead. By splitting this onto multiple processes distributed across different compute nodes we gain access to sufficient memory to fill very large domains and three dimensional domains. An example of such a simulation is shown in [figure 4.4](#)

4.2 Postprocessing

The SPH code outputs checkpoint files to allow resumption of interrupted simulations. These files are a serialised representation of the internal state of the simulation code and as such contain much information which is not needed for analysing the simulation results. For example, they contain the positions, velocities etc. at the intermediate timestep used in the timestepping scheme⁴. For this reason the postprocessing software converts the checkpoint files to VTK format for visualization with Paraview or analysis with Python and its VTK bindings.

As well as converting the SPH particle data, the postprocessor also performs remeshing and isosurface calculations. An isosurface, the 3D equivalent of a contour, is the surface on which a quantity has a constant value. By producing an isosurface for the number density of the fluid phase we can reconstruct the fluid interface and visualize the flow more easily. [Figure 4.5](#) shows the results of a 3D multiphase simulation where two phases, air and water, have been simulated. The water phase has been visualized in three different ways; SPH particles, an isosurface and the isosurface coloured according to the water velocity.

In order to calculate the isosurface for a particular fluid phase the number density for that phase

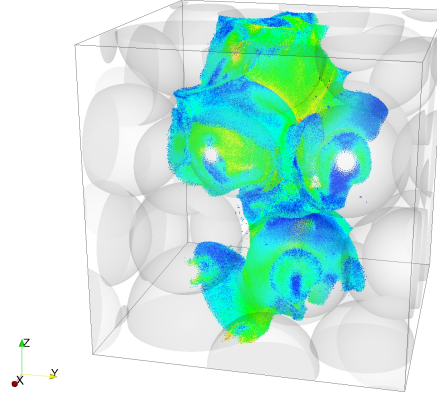
$$\sigma_l(\mathbf{x}) = \sum_{j \in \text{phase } l} W_{ij} \quad (4.4)$$

is evaluated at the points of a regular cubic grid with spacing Δx . The marching cubes algorithm is then applied to calculate the corresponding isosurface. Isosurfaces have the advantage over the particle representation in that, because their normals can be calculated, it is possible to apply shading which makes it easier for the eye to understand depth and shape cues. Further, using the SPH interpolation function [equation \(3.1\)](#) the other quantities such as velocity and pressure can be interpolated onto the surface allowing us to colour the surface by quantities of interest as shown in [figure 4.5c](#) where the surface has been coloured according to the water's velocity.

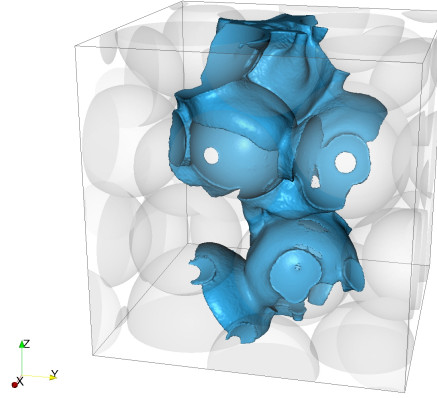
4.3 Periodic Packings

Because the domains are periodic we also require that the REV geometries are periodic to avoid wall effects or having unrealistic channels in the geometries at the periods. The generation of

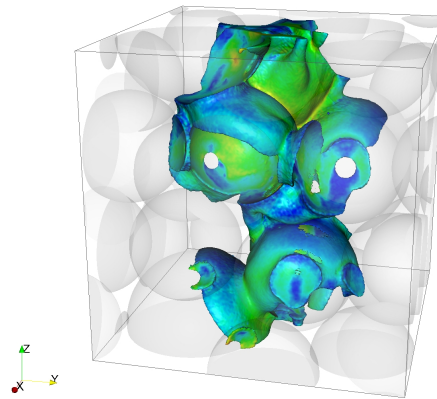
⁴See [section 3.4](#).



(a) Particles



(b) Isosurface



(c) Coloured Isosurface

Figure 4.5: Example visualizations of a 3D simulation. [Figure 4.5a](#) shows the SPH particles coloured by velocity, [figure 4.5b](#) shows the density isosurface with shading and [figure 4.5c](#) shows the same isosurface coloured according to the remeshed velocity of the fluid.

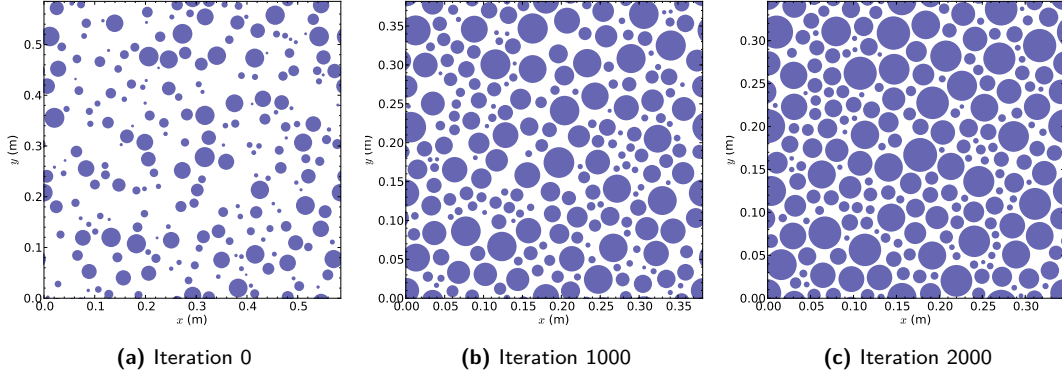


Figure 4.6: The progress of the periodic packing algorithm shown at various iterations. Note that in 2D a gap is left between the particles so that fluid can flow, hence the padding between the particles.

the periodic packings used in this thesis was accomplished by using a simple DEM type scheme. The algorithm is initialized by placing a number of spherical (or circular) particles into a large periodic domain such that they do not overlap. Once this step is complete the domain is shrunk by an amount $\Delta L \ll L$ according to the affine transformation

$$\mathbf{x} \rightarrow \mathbf{x} \left(1 - \frac{\Delta L}{L} \right) \quad (4.5)$$

where L is the domain length. This transformation is applied to every point within the domain meaning that the positions of each circle (sphere) is also updated. The radii, however, remain unchanged by this operation. Because the radii are unchanged, once this transformation has been applied some particles may now be overlapping. Any overlapping particles feel a repulsive force which is given by

$$\mathbf{f}_{ij} = \frac{1}{2} \lambda_{ij} \hat{\mathbf{e}}_{ij}$$

where λ_{ij} is the overlap of particles i and j given by

$$\lambda_{ij} = \max \{ 0, (R_i + R_j + p) - |\mathbf{d}\mathbf{x}(\mathbf{x}_i, \mathbf{x}_j)| \},$$

p is a fixed padding width which lets us leave a gap between the particles. If $p = 0$ in 2D then nothing could flow in the system. The total force felt by a particle is the sum of forces due to overlap and a drag force;

$$\mathbf{f}_i = -\eta |\mathbf{v}_i|^2 + \sum_j \frac{\lambda_{ij}}{2} \hat{\mathbf{e}}_{ij}$$

where the drag coefficient η is set to 0.001 kg s^{-1} . Since the force is only linear in the overlap the particle interactions are relatively ‘soft’ but this is not an issue here since we are not interested in the dynamics of the system, just the final configuration of particles in the periodic domain.

The system is evolved in time using the improved Euler time stepping scheme, until it has settled and no particles are overlapping. Then the domain is shrunk once more. This cycle

continues until the domain cannot be shrunk any more. All particles are assumed to have the same density ρ so the mass of each particle depends only on its radius R_i . The radii of the particles are chosen from a particular distribution depending on the simulation.

4.4 Conclusions

This chapter has detailed the issues associated with the actual computation underlying the SPH method and the analysis of the results. Translating the concepts detailed here into code was itself a large task, the code base for the SPH simulation alone is over 30,000 lines of C++.

Central to the SPH code is the intrusive multi-list data structure in which the SPH particles are stored. This design is motivated by the simultaneous needs to be able to iterate over the SPH particles in different ways (applying a function to each particle and finding nearby particles) and keep the memory used clustered together to improve performance. Using a list like structure also allows for quick reordering & removing of particles and combining or splitting lists as is necessary at various points in the code. The implementation of a ‘store’ list also helps to improve performance by removing repeated allocation and deallocation of memory and helping to minimize memory fragmentation.

The linked cell grid is an established method in the literature [41] but it is not the only choice. Many SPH codes use tree structures, such as a $k-d$ tree, for neighbour lookups. Tree structures have performance which scales as $O(\log N)$ and thus is slower than the linked cell grid’s $O(1)$ neighbour search. However, in simulations with long range forces one is limited to these slower searches as one cannot cut off the interactions beyond a certain length. Because in our systems we have only the hydrodynamic interactions, which are short range, the choice of a linked cell grid is clear.

Chapter 5

Wall Boundaries

In flows through packed beds the interaction of the fluid with the solid surfaces is of vital importance, both through the effect of partial wetting (and contact angles) and due to the shearing of fluid which takes place there. Clearly the ability to accurately capture wall boundaries and associated effects in our SPH simulations is necessary. SPH was originally developed for application to astrophysical gas flows [40, 72] where boundaries are not required. As applications shifted from astrophysics to engineering simulations, several methods have been used to include solid walls in SPH simulations. Many of these, however, are not suitable at the relatively small scales seen in our interstitial flows. In addition to capturing the fluid behaviour near walls accurately, we must also be able to handle the complex shapes arising in the packed beds seen in heap leaching.

A recent investigation by Cummins et al [27] found that of the boundary formulation, the kernel used, kernel correction, time-stepping scheme and the compressibility of the SPH fluid, the boundary formulation had the largest effect on the simulation results. The earliest (and still widely used) are repulsive boundary conditions [60, 80, 83, 86], in which a repulsive force is applied to any particles which approach the wall boundary. Typically the repulsive force takes the form of a truncated Lennard-Jones type force. The advantages of this type of boundary is that it is simple to implement and is good at ensuring particle containment within the domain.

One disadvantage, however, is that due to the stiff nature of the Lennard-Jones potential it can require a very small time-step to ensure stability. Further, because it acts only normal to the wall, it cannot correctly capture shear stresses due to velocity gradients near the wall. This becomes an issue at smaller scales or higher viscosities. Worse still, because the forces only ever act outwards from the boundaries, they cannot reproduce any system under tension, such as a hanging droplet, for instance.

The support of smoothing kernels for particles near the boundaries extends out of the domain leading to so-called ‘kernel boundary deficiency’. The deficiency leads to misestimation of quantities near the boundaries. While repulsive boundaries contain the particles in the domain they do not attempt to remedy this problem, unlike so-called ‘ghost particle’ boundaries.

Ghost particles are SPH particles which sit outside the domain, up to a distance of $2h$, thereby eliminating the kernel support deficiency. The ghost particles interact with the fluid particles through the normal pressure and viscosity forces. This method is preferable to the repulsive boundaries if the ability to capture shear-stresses and tension are needed. Since the interaction with the fluid particles is through the usual equations of motion the time-step is not unduly affected. There are two different ghost particle methods; *static* and *reflected*.

For the static type, the ghost particles are placed on a regular grid outside the fluid domain and their positions remain fixed for the entirety of the simulation (barring the movement of any

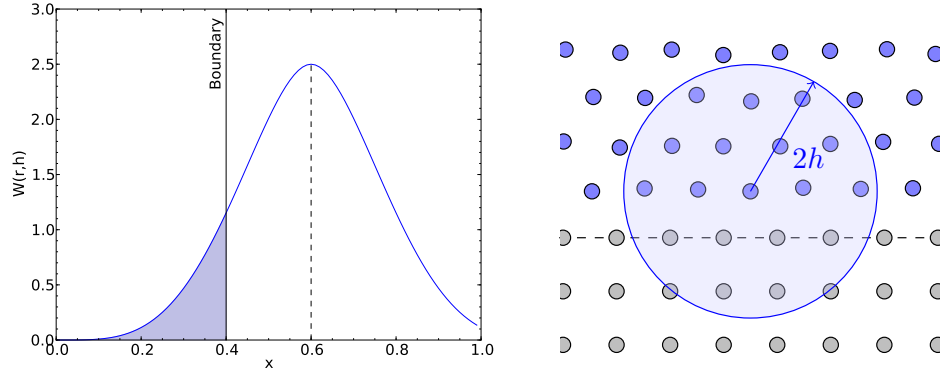


Figure 5.1: When an SPH particle is near the domain boundary, if nothing extra is done, its support extends beyond the domain leading to *kernel boundary deficiency* which leads to misestimation of quantities near the boundary. This can be corrected by placing *ghost particles* to a width of $2h$ outside the domain.

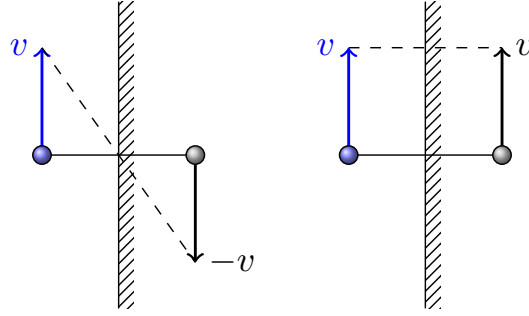


Figure 5.2: By reflecting the fluid particles across a boundary and setting their velocities appropriately one can capture the required no-slip of free-slip boundary conditions.

boundaries with time). Examples of their application can be found in [28, 84]. This method is useful as it is computationally efficient and is a simple extension to any SPH code. However, while it enforces some shear at the boundaries it is not capable of properly enforcing no-slip conditions. To see why this is the case consider the smoothed velocity field $\langle \mathbf{u} \rangle(\mathbf{x})$. At a boundary the smoothing operation sees the fluid with a linear velocity profile, but on the other side the ghost particles, by definition, have velocity zero. This means the averaged velocity at the boundary is non-zero. Static ghost particle also cannot capture free slip conditions in their usual form.

To counter this non-zero smoothed velocity at the boundaries, one can use reflected ghost-particles [49, 82]. The method works much like the static ghost-particle method, i.e. by extending the computational domain by $2h$ outwards to overcome the lack of kernel support, but instead of static ghost particles the fluid particles are reflected across the wall-boundary at each time-step. The velocity of each particle is also reflected for the ghost particle to become $-\mathbf{u}$, thereby ensuring that the smoothed velocity field $\langle \mathbf{u} \rangle$ goes to zero at the domain boundaries.

Alternatively the reflected particles can also be given the same velocity as their image particle

thereby leading to free slip boundary conditions. The two cases are illustrated in [figure 5.2](#). The reflection operation requires slightly more computational effort but it ensures the velocity profile is correctly reproduced at the wall.

Recent work has taken another approach which involves correcting the kernel boundary deficiency without extending the domain [\[36, 37, 63\]](#). These look a promising way to capture boundary behaviour and are an active area of development, see [section 5.4](#).

5.1 Geometric Objects

While faceted geometries present a very general and portable solution to including wall-boundaries in SPH simulations, allowing complex and arbitrary shapes, they are not without issues [\[6\]](#). In many situations simplified or idealized geometries suffice to elucidate the salient elements of the flows at hand. For this reason we also consider the inclusion of basic geometric objects into our SPH code.

By geometric objects we mean objects whose shapes are easily representable as equations, such as spheres, cylinders, torii, etc. Such objects have several advantages over faceted surfaces; firstly the only values which need to be stored in memory are the relevant parameters. For instance for a sphere we need only store four floating point numbers, the radius R and the 3D centre $\mathbf{r}_c$. While not generally an issue for CPU based implementations this could be considered a benefit for GPU based codes which have access to less RAM.

The second, and primary, advantage is that such simulations are robust to changes in resolution. For a complex faceted geometry it is often necessary to coarsen an initially fine mesh so that the extent of the facets roughly matches the resolution of the simulation. If the resolution then changes, further processing might be required. Since geometric objects are purely defined by their parameters this is clearly not an issue.

5.1.1 Curved Boundary Correction

These geometric objects will in general have curved surfaces, a fact which must be taken into account to ensure the correct interpolation of quantities over the boundary. However this fact has not been addressed in the literature despite papers using curved boundaries [\[86, 105\]](#).

Consider reflecting a particle in a circular boundary with radius R and centre $\mathbf{r}_c$. The signed distance of the particle to the circle is d . The sign of d indicates whether we are inside the circle or not; $d > 0$ means inside and $d < 0$ means outside. If the fluid particle is at $\mathbf{r}$ then $d = R - |\mathbf{r} - \mathbf{r}_c|$ and the reflected particle will be placed at

$$\mathbf{r}' = \mathbf{r} + \frac{(\mathbf{r} - \mathbf{r}_c)}{|\mathbf{r} - \mathbf{r}_c|} 2d. \quad (5.1)$$

Assuming a regular spacing of SPH particles we would expect that around the circle with radius $R - d$ there would be $2\pi(R - d)/\Delta x$ particles, where Δx is the average particle spacing in the domain. This means that around the reflected circle there will also be $2\pi(R - d)/\Delta x$

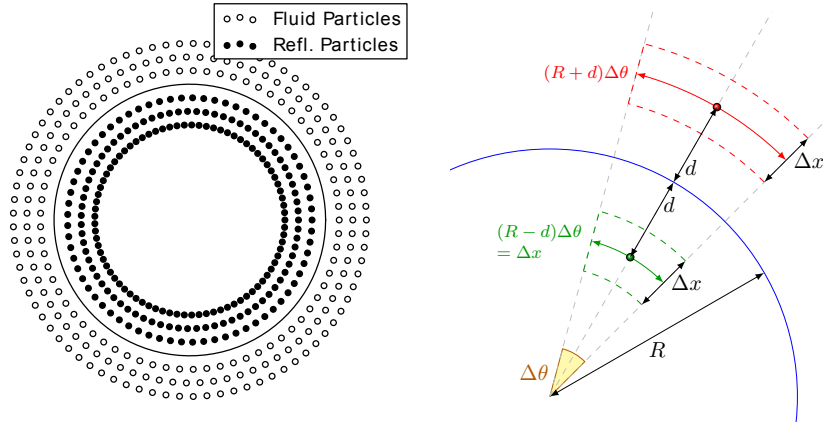


Figure 5.3: An illustration of how reflecting regularly spaced fluid particles in a curved surface leads to an increase in particle number density if reflecting inwards, or a decrease if reflecting outwards and a simple geometric construction showing the 'volume factor' associated with such a reflection.

reflected particles. However the reflected circle has a radius of $R + d$ so we would instead expect $2\pi(R + d)/\Delta x$ particles to be present if the particles were regularly spaced. Thus for each reflected particle expected we actually have $(R - d)/(R + d)$ reflected particles, meaning each reflected particle represents a different volume than its matching fluid particle. An illustration of this is shown in [figure 5.3a](#).

The ratio of the reflected particle j 's volume to its matching fluid particle's volume is

$$\beta_j = \frac{R + d}{R - d}. \quad (5.2)$$

This can be written in terms of the curvature $\kappa = 1/R$:

$$\beta_j = \frac{1 + \kappa d}{1 - \kappa d}. \quad (5.3)$$

Writing β_j in this form allows us to apply this correction to any arbitrary shape for which we know the curvature at the reflection point. This generalizes to three dimensions as

$$\beta_j^{3D} = \left(\frac{1 + \kappa_1 d}{1 - \kappa_1 d} \right) \left(\frac{1 + \kappa_2 d}{1 - \kappa_2 d} \right), \quad (5.4)$$

where κ_1 and κ_2 are the two principle curvatures of the surface at the closest point to the fluid particle. The volume factor for a number of shapes is shown in [table 5.1](#).

Particle spacing is accounted for by σ which is inversely proportional to the volume V . Because curved boundaries lead to either more or fewer reflected particles than would otherwise be expected, we weight the contribution to the fluid particle's σ value from reflected particles by β . This gives a slightly modified form for calculating σ ;

$$\sigma_i = \sum_j \beta'_j W_{ij}. \quad (5.5)$$

Table 5.1: Volume factor correction, β_j , for a number of geometric shapes.

Shape	d	β	Symbols
Circle	2	$\frac{R+d}{R-d}$	R - Radius
Sphere	3	$\left(\frac{R+d}{R-d}\right)^2$	R_1 - Wheel radius R_2 - Tube radius, $R_2 < R_1$
Cylinder	3	$\frac{R+d}{R-d}$	θ - Poloidal angle, with 0 being inside.
Torus	3	$\left(\frac{R_2+d}{R_2-d}\right) \left(\frac{R_1-R_2-d}{R_1-R_2+d}\right) \cos \theta$	

where

$$\beta'_j = \begin{cases} \beta_j & \text{if } j \text{ is a reflected particle} \\ 1 & \text{otherwise} \end{cases}, \quad (5.6)$$

This ensures σ_i is calculated correctly for fluid particles which are within $2h$ of the boundary.

For reflected particles, σ_i must be copied from the particle from which it was reflected. This is because we cannot calculate σ at the reflected location due to the lack of kernel support. If we are reflecting in a straight boundary simply setting $\sigma_{\text{refl}} = \sigma_{\text{fluid}}$ is correct because the σ 's for the reflected and matching fluid particles will be equal. For curved boundaries however we must weight the reflected particles' σ 's because the spacing of the reflected particles is different from that of the fluid particles from which σ is being copied. The correct weighting is again given by the volume factor:

$$\sigma_{\text{refl}} = \sigma_{\text{fluid}} \beta^{-1}. \quad (5.7)$$

The correction becomes less important as the spatial resolution is increased. Since we expect $d \approx \Delta x$ (or $\Delta x/2$ if not using boundary particles) as we decrease Δx we have

$$\lim_{d \rightarrow 0} \beta = 1 \quad (5.8)$$

as expected, however results show that even for high resolutions this correction still confers a benefit over simply reflecting.

5.1.2 Example: Couette Flow

An example where curved boundaries are useful is that of a rotating flow between two cylinders, known as Couette flow. In this example a fluid with density $\rho = 1000 \text{ kg m}^{-3}$, dynamic viscosity $\mu = 0.001 \text{ Pa s}$ is contained between two cylinders of radius $R_i = 2.5 \text{ cm}$ and $R_o = 4.0 \text{ cm}$. The inner cylinder rotates with angular velocity ω . For the hydrostatic case where $\omega = 0 \text{ s}^{-1}$ [figure 5.4](#) shows the relative density at the zeroth timestep of two simulations, one with the volume correction factor applied and one without. It is clear that the volume correction factor has improved the density estimate significantly.

After a time $T = 1.0 \text{ s}$ ([figure 5.5](#)) in the volume corrected simulation the particles maintain the correct spacing, whereas without this correction factor the particles rearrange themselves

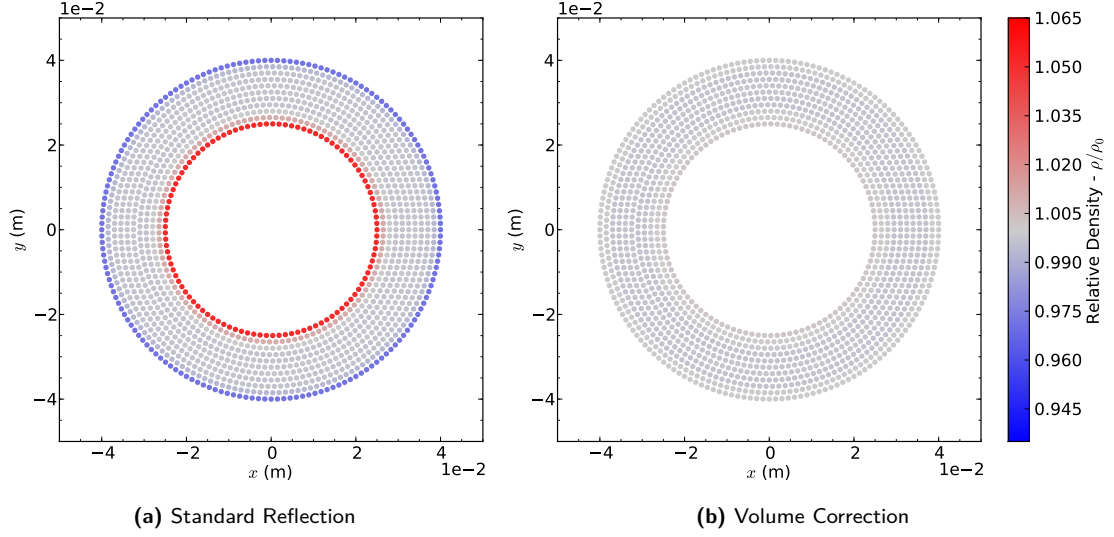


Figure 5.4: The calculated density field at time $t = 0$ using only reflection (5.4a) and reflection with volume correction (5.4b). It is clear that the density is better estimated when using β .

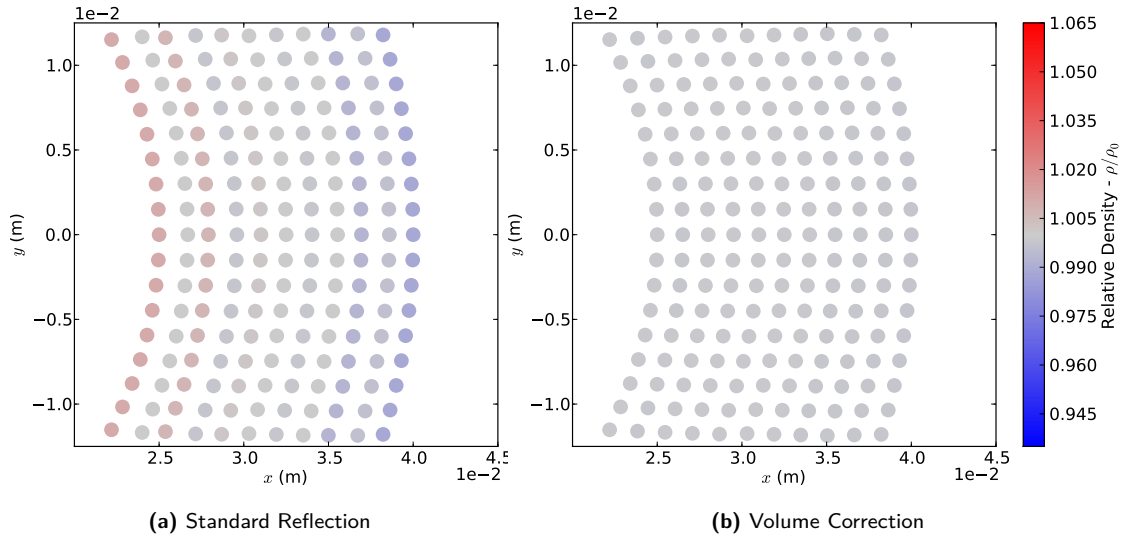


Figure 5.5: The calculated density field after $t = 1.0s$ using only reflection (5.5a) and reflection with volume correction (5.5b). It is clear that the density is better estimated when using β and that the particles correctly maintain their spacing.

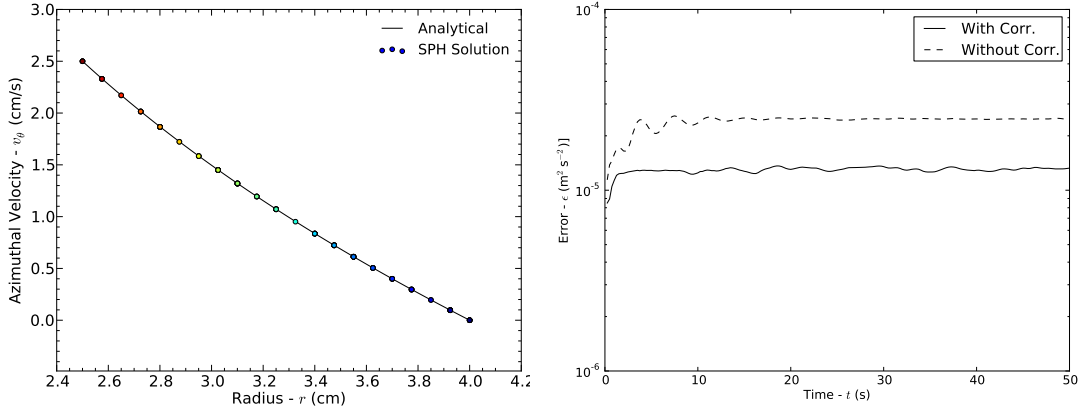


Figure 5.6: 5.6a shows the analytical solution for the Couette flow compared to the SPH solution. Excellent agreement is seen. 5.6b shows the error as a function of time for the Couette flow. The solid line represents the solution with volume-factor correction and the dotted line without. It is apparent that the correction improves the solution markedly.

to attempt to even out the density. Even with this rearrangement the density has not fully corrected itself by 1 s; the particles near the inner wall are pushed away whereas the particles near the outer wall are pulled towards it. The effect is smaller for the outer wall simply because it has a lower curvature.

Next consider the case where the inner cylinder rotates with angular velocity $\omega = 1 \text{ s}^{-1}$. At steady state the analytical velocity profile as a function of radius - r - is

$$v_r = 0, \quad v_\theta = \frac{\omega}{1 - \eta^2} (R_i^2 r^{-1} - \eta^2 r), \quad (5.9)$$

where $\eta = R_i/R_o$, see [100]. The error of the SPH solution is quantified by the root-mean-square (RMS) deviation from the analytical solution

$$\epsilon = \sqrt{\frac{1}{N} \sum_{i=1}^N (v_{\theta,i} - v_\theta(r_i))^2}, \quad (5.10)$$

where the sum is over SPH fluid particles only, i.e. not including boundary particles. Figure 5.6b shows the error in the solution against time for both the standard reflection and volume-corrected reflection. It is clear that the correction leads to large improvement in the solution, demonstrating that this is not something which can be ignored.

Figure 5.7 shows the error for both corrected and uncorrected reflection as a function of the spatial discretization length Δx . It shows that, as expected, the correction makes less difference as we increase the resolution. But even for the very highest resolution, where the simulation has >85,000 particles, the correction still provides a noticeable improvement to the solution.

Since the domain (and hence fluid flow) for this example is curved, a pressure gradient is set up which depends on r . Because of this pressure gradient a small background pressure of

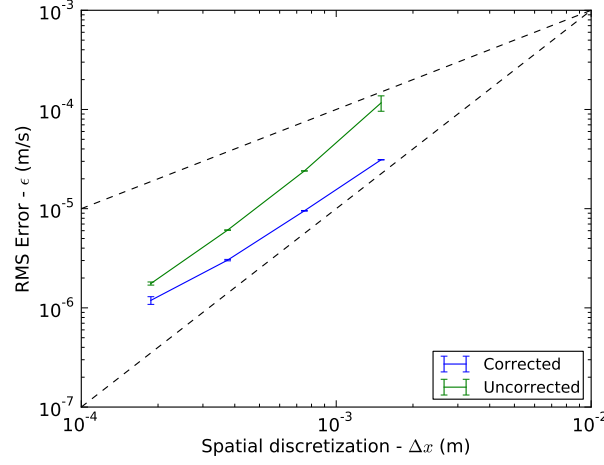


Figure 5.7: The error in the SPH solution as a function of spatial discretization length for both the simple reflection and the corrected reflection. One can see that the correction significantly improves the solution at all resolutions

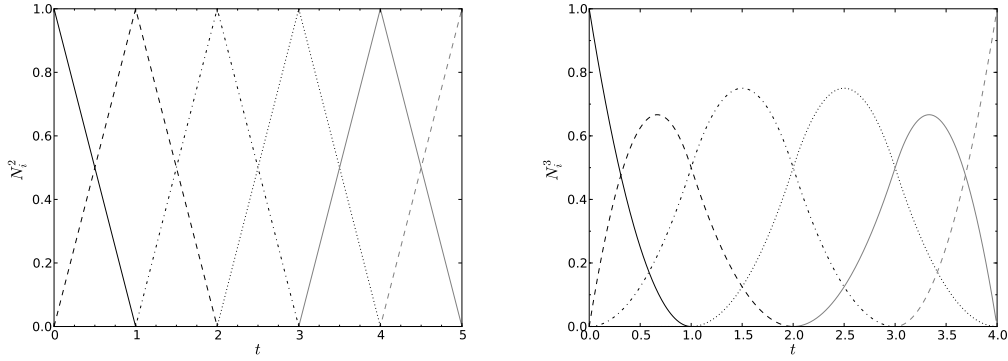


Figure 5.8: B-spline basis functions N_i^k for $k = 2, 3$ with the non-periodic knot vector $\mathbf{T} = [000123444]$.

$P_0 = 2.5$ Pa is added to improve the stability and avoid tensile instability. For consistency this pressure gradient was kept constant at 2.5 across all Δx , which leads to a slightly strange rate of convergence since at higher resolutions the pressure required to avoid tensile instability actually decreases.

5.2 B-Splines Curves

Non-uniform rational B-splines (NURBSs) are a superset of the perhaps better known Bezier curves. They have become well established as the *de-facto* standard for representing and drawing curves and surfaces in CAD packages. The ability to natively include them in a CFD simulation would be therefore very useful.

A NURBS curve is defined by its control points $P = \{\mathbf{P}_i, i = 1 \dots n\}$, their weights $\Omega = \{w_i, i =$

$1 \dots n\}$ and the so-called knot vector $\mathbf{T} \in \mathbb{R}^{k+n}$. The curve is then parameterized as

$$\mathbf{C}(t) = \frac{\sum_{i=1}^n w_i N_{i,k}(t) \mathbf{P}_i}{\sum_{i=1}^n w_i N_{i,k}(t)} = \sum_{i=1}^n R_{i,k}(t) \mathbf{P}_i, \quad (5.11)$$

where k is the order of the B-spline interpolation, $k = 1$ being piecewise constant, 2 being piecewise linear, and so on. Where we have set

$$R_{i,k}(t) = \frac{w_i N_{i,k}(t)}{\sum_{j=1}^n w_j N_{j,k}(t)} \quad (5.12)$$

for brevity. The B-spline basis function of order k for control point i is defined by the recursive relationship [25, 31]

$$N_{i,k}(t) = N_{i,k-1}(t) \frac{t - T_i}{T_{i+k-1} - T_i} + N_{i+1,k-1}(t) \frac{T_{i+k} - t}{T_{i+k} - T_{i+1}} \quad (5.13)$$

for $k > 1$. For $k = 1$ we have

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } t \in [T_i, T_{i+1}) \\ 0 & \text{otherwise} \end{cases} \quad (5.14)$$

The basis functions satisfy the partition of unity property meaning $\sum_i N_{i,k}(t) = 1 \forall k, t$. A similar relation holds for defining a NURBS surface in 3D as a function of two parameters

$$\mathbf{S}(t, u) = \frac{\sum_{i=1}^p \sum_{j=1}^q N_{i,k}^1(t) N_{j,l}^2(u) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=1}^p \sum_{j=1}^q N_{i,k}^1(t) N_{j,l}^2(u) w_{i,j}}, \quad (5.15)$$

where in 3D the control points have the form of a regular grid with the number of points in the t and u directions respectively being p and q . The basis functions N^1 and N^2 are different in the two directions only by virtue of having possibly different knot vectors. Figure 5.8 shows an example of the basis functions $N_{i,k}$ for $k = 2$ and 3.

For ease of discussion in all examples presented here the weights w_i are set equal to unity. This simplifies the maths but retains the key aspects for application to SPH. For the full derivations and expressions we refer the interested reader to [88] for example.

5.2.1 Gradients and Curvature

Because B-Spline curves are defined mathematically it is easy to calculate quantities such as normal vectors, tangents and curvatures given a parameter value t . The gradient of the curve $\mathbf{C}(t)$ is just the tangent vector and is given by

$$\frac{d\mathbf{C}}{dt} = \sum_{i=1}^n N'_{i,k}(t) \mathbf{P}_i := \mathbf{t}(t), \quad (5.16)$$

with

$$N'_{i,k}(t) = \frac{N_{i,k-1}(t) + (t - T_i)N'_{i,k-1}(t)}{T_{i+k-1} - T_i} + \frac{(T_{i+k} - t)N'_{i+1,k+1}(t) - N_{i+1,k-1}(t)}{T_{i+k} - T_{i+1}}, \quad (5.17)$$

for $k > 2$. For $k = 2$

$$N'_{i,2}(t) = \frac{N_{i,1}(t)}{T_{i+1} - T_i} - \frac{N_{i+1,1}(t)}{T_{i+2} - T_{i+1}}. \quad (5.18)$$

Similarly the second derivative is

$$\frac{d^2 \mathbf{C}}{dt^2} = \sum_{i=1}^n N''_{i,k}(t) P_i. \quad (5.19)$$

with

$$N''_{i,k}(t) = \frac{2N'_{i,k-1}(t) + (t - T_i)N''_{i,k-1}(t)}{T_{i+k-1} - T_i} + \frac{(T_{i+k} - t)N''_{i+1,k+1}(t) - 2N'_{i+1,k+1}(t)}{T_{i+k} - T_{i+1}} \quad (5.20)$$

for $k > 3$. For $k = 3$

$$N''_{i,3}(t) = 2 \left(\frac{N'_{i,2}(t)}{T_{i+2} - T_i} - \frac{N'_{i+1,2}(t)}{T_{i+3} - T_{i+1}} \right). \quad (5.21)$$

In two dimensions the normal to the curve can be found by taking the cross product with a unit vector pointing along the third (z) axis, i.e.

$$\mathbf{n}(t) = \frac{\mathbf{t}(t) \times \hat{\mathbf{z}}}{|\mathbf{t}(t)|}, \quad (5.22)$$

and the curvature at any point is given by

$$\kappa(t) = \frac{|\mathbf{t}(t) \times \mathbf{C}''(t)|}{|\mathbf{t}(t)|^3}. \quad (5.23)$$

5.2.2 Reflecting in B-Spline Curves

There are several computational factors to consider before using NURBS with reflected boundaries in SPH. Foremost amongst these is the ability to quickly reflect the fluid particle accurately across the NURBS curve or surface. To perform this operation we need to know the normal, closest point to the surface and, to properly account for the curved nature of these boundaries, the curvature. Given the parameter value which corresponds to the closest point we can calculate all these auxiliary quantities directly from the analytical forms (5.22), (5.11) and (5.23). The difficulty lies in quickly yet accurately calculating the value $t_{\min}$ which locates the closest point on a curve to an arbitrary point $\mathbf{x}$ in $\mathbb{R}^d$.

As there is a stationary layer of particles along the boundary (see section 4.1.3) we can utilize these to quickly locate $t_{\min}$. Each boundary particle stores the value of t which corresponds to its location. Then, when reflecting a fluid particle, a nearby boundary particle can be located quickly using the linked-cell-grid method used in SPH codes and its value of t is taken as an initial guess for $t_{\min}$.

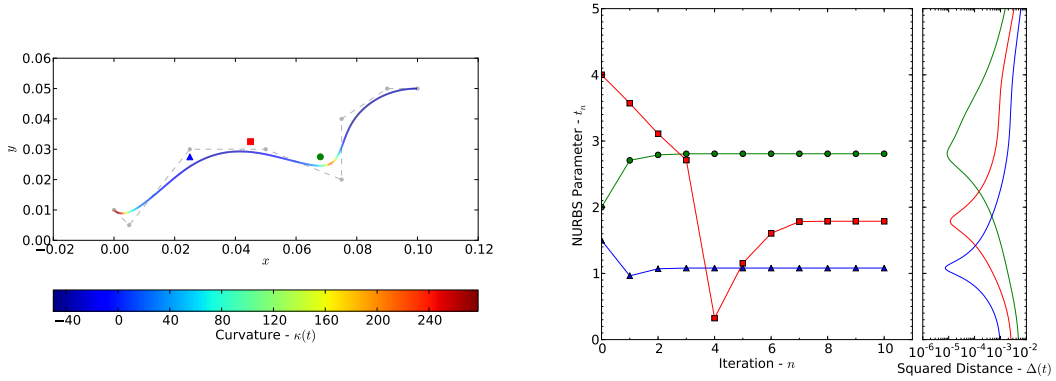


Figure 5.9: An example NURBS curve showing the convergence of the Newton-Raphson method for locating the closest point on the curve $\mathbf{x}_{np}$ to an arbitrary point $\mathbf{x}$. Convergence is shown for three different points indicated by $\bullet$, $\blacksquare$ and $\blacktriangle$.

We use the Newton-Raphson method to find the minimum of the function Δ with respect to t .

$$\Delta(t) = \|\mathbf{x} - \mathbf{C}(t)\|^2 \quad (5.24)$$

$$= (\mathbf{x} - \mathbf{C}(t)) \cdot (\mathbf{x} - \mathbf{C}(t)). \quad (5.25)$$

$\Delta(t)$ gives the squared distance between a point $\mathbf{x}$ and the point on the NURBS curve defined by parameter value t .

To perform Newton-Raphson optimization we need to know both the first and second derivatives of Δ . By expanding the dot product in (5.25) it is simple to show that

$$\frac{d\Delta}{dt} = 2(\mathbf{C} - \mathbf{x}) \cdot \frac{d\mathbf{C}}{dt}. \quad (5.26)$$

The quantity $\frac{d\mathbf{C}}{dt}$ represents the tangent vector to the curve at t , so at any extrema the fact $\frac{d\Delta}{dt} = 0$ implies that the vector between the point $\mathbf{x}$ and the extremum is normal to the curve as expected. By differentiating again we get the second derivative

$$\frac{d^2\Delta}{dt^2} = 2 \left[\mathbf{t} \cdot \mathbf{t} + (\mathbf{C} - \mathbf{x}) \cdot \frac{d^2\mathbf{C}}{dt^2} \right]. \quad (5.27)$$

The iterative sequence for optimizing t is given by

$$t_{n+1} = t_n - \frac{\Delta'(t_n)}{\Delta''(t_n)}, \quad (5.28)$$

which, using the results above, is

$$t_{n+1} = t_n - \frac{(\mathbf{C} - \mathbf{x}) \cdot \mathbf{t}}{\mathbf{t} \cdot \mathbf{t} + (\mathbf{C} - \mathbf{x}) \cdot \frac{d\mathbf{t}}{dt}}. \quad (5.29)$$

The iterative scheme given by [equation \(5.29\)](#) is only useful if it converges quickly. In typical SPH simulations we find that, with a tolerance of $0.01\Delta x$, [equation \(5.29\)](#) always converges within 10 iterations and usually with as few as 3. This is partly due to the strength of the Newton-Raphson method for the function $\Delta(t)$ and partly due to the initial guesses for t_0 being close to $t_{\min}$.

However, certain situations can lead to difficulty for the location of $t_{\min}$. If there are areas of high positive curvature there can exist two local minima for the distance function. Additionally for certain choices of t_0 the method diverges. There are two possible causes for this; either it is spurious or the closest point is truly one of the end points. This can be checked by comparing t_n to our initial guess t_0 . Since we know t_0 must be close to the true value for $t_{\min}$ we check to see if our point is close to either end. If this is the case we can assume the method has not spuriously diverged.

If, however, we hit either $t = 0$ or $t = (N - K + 1)$ and t_0 is not close to either end we assume this divergence is spurious, move t_0 slightly and then try again. In these situations we define t_0 to be close to the endpoints if it is within $(N - K + 1)/(2N)$ of either limit. If there are control points bunched near either end point this might be a poor definition, but in practice it was found to be adequate to distinguish the cases.

An additional check which was found to improve stability is to look at the gradient. For all cases except those of high curvature, the function $\Delta(t)$ is observed to be quasiconvex meaning we always wish to be moving in the direction of negative gradient. If a Newton-Raphson step is hit which would move up the gradient it is instead moved the same distance in the direction of negative gradient. [Figure 5.9](#) shows the convergence of the Newton method for three example points near the curve. For the red curve it over shoots but is still able to converge within 10 iterations. Additionally, in these examples the initial guess for t is further from $t_{\min}$ than is realistic for an SPH simulation.

The cost of calculating $t_{\min}$ in this manner does not depend in any way on the number of particles in the system N but does depend slightly on the number of control points for the B-spline curve n , because calculating the point on a curve requires a summation involving each control point. Thus for each particle which is near a B-spline boundary the calculation of $t_{\min}$ represents a small fixed cost. Once $t_{\min}$ has been calculated the reflection position and curvature are all easily calculable in $O(n)$ time. Thus the overall method remains $O(N)$ in the number of particles. Overall the performance of the SPH code was not noticeably affected by the inclusion of B-spline boundaries.

5.2.3 Concave Corners & Piecewise Curves

It should be noted that according to [equation \(5.3\)](#) if the radius of curvature approaches Δx and the signed distance is $+\Delta x$, then $\beta \rightarrow \infty$. This is the situation corresponding to a sharp concave corner (from the point of view of the fluid). This case of high positive curvature can also cause issues for the convergence of $t_{\min}$ as detailed above. For these two reasons sharp corners in the B-spline curves should be avoided. By separating the boundary at a sharp corner

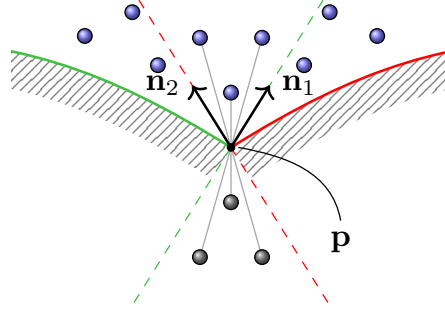


Figure 5.10: By checking the normals of the two boundaries, valid nearby SPH particles can be reflected in the point $\mathbf{p}$ to fill the gap that would otherwise be present.

into two pieces one can avoid this issue. Having two curves meet at an angle which is less than 180 degrees leads to an area where, if particles were only reflected in the B-Splines, there would be a lack of SPH particles. This can be remedied easily by utilizing point symmetries and reflecting nearby particles across the point at which the two walls meet, as illustrated in figure 5.10.

One must take care that only the particles whose reflections lie in the gap are actually reflected. To check this one can examine the boundary normals, $\mathbf{n}_1$ and $\mathbf{n}_2$ at the point $\mathbf{p}$. If the following inequality holds

$$(\mathbf{n}_1 \times (\mathbf{r} - \mathbf{p}))_z \cdot (\mathbf{n}_2 \times (\mathbf{r} - \mathbf{p}))_z > 0, \quad (5.30)$$

then the particle whose position is $\mathbf{r}$ lies in the valid region to be reflected. This holds, of course, for straight line segments as well as curved boundaries. Similar checks can be performed where two boundaries meet at an angle greater than 180 degrees to avoid reflecting two particles into the same region, but this is not necessary with B-spline boundaries since one can introduce sharp convex angles without encountering the issues above. This is because $\beta \rightarrow 0$ as the radius of curvature approaches Δx for negative curvatures.

It should be noted that a particle which is reflected into the gap at one timestep may not be reflected into the gap at the next timestep even if the distances is less than $2h$, which causes an abrupt change in the arrangements of reflected particles between two consecutive timesteps. If approximating a curve as a series of line-segments the corners must be handled as described here; meaning that at any corner we would have issues with gaps. This is not necessary for geometric objects except at the end or intersections of any curves thereby removing another issue with approximating curves as a series of line segments.

5.2.4 Placing Boundary Particles

The boundary particles must be placed at regular intervals of Δx along the curve. If the control points for the B-spline curve are all the same distance apart, and the knot vector intervals are equal (i.e. $T_{i+1} - T_i = T_{i+2} - T_{i+1}$ for all $i = 1, \dots, (N + K - 3)$) then moving a set distance in t will move a set amount along C in $\mathbb{R}^d$ at all points along C . However, this is quite a

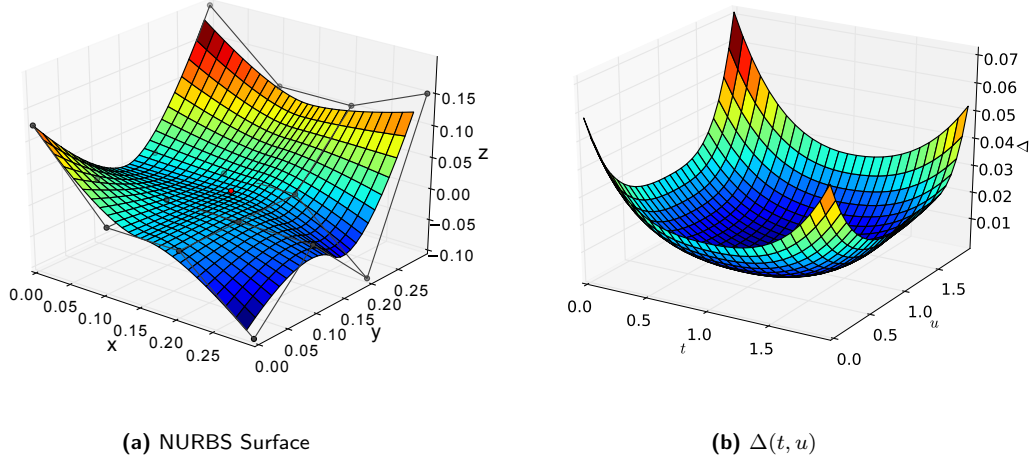


Figure 5.11: An example of a NURBS surface $S(t, u)$ in 3D. It has 16 control points and $K = 3$. Also shown in the squared distance from the red point $\Delta(t, u)$ as a function of the two parameter values.

strict requirement on the curve which would significantly reduce its usefulness. For this reason we numerically integrate the arc-length $s(t)$ along the curve and place boundary particles at regular intervals.

$$s(t) = \int_0^t \left. \frac{d\mathbf{C}}{dt'} \right|_{t'=t} dt' \quad (5.31)$$

$$\approx \sum_{i=1}^n |\mathbf{C}_i - \mathbf{C}_{i-1}| (t_i - t_{i-1}). \quad (5.32)$$

While this is a relatively slow operation, it only needs to be performed once at the very beginning of the simulation and placing boundary particles in this manner removes any restrictions on the control-points and knot-vector.

5.2.5 Application in Three Dimensions

The examples and methods shown in this thesis focus on 2D NURBS curves, but, as mentioned above, it is also possible to define NURBS surfaces in 3D. The Newton-Raphson method used to locate closest points also works for a two-dimensional parameter space. The function Δ is still well behaved for a NURBS surface meaning the Newton-Raphson method should be easily applicable to 3D simulations. Figure 5.11b shows the behaviour of Δ over the parameter space for the example NURBS surface shown in 5.11a.

The larger issue in 3D comes when placing the boundary particles. In 2D it is possible to simply numerically integrate the arc length along the curve with a fine step size and place particles at regular intervals of Δx . Because the mapping between (t, u) and the position on the surface is non-trivial it is difficult to say what pattern one must follow in (t, u) space to give an even distribution of boundary particles. This does not mean that the NURBS surfaces cannot be

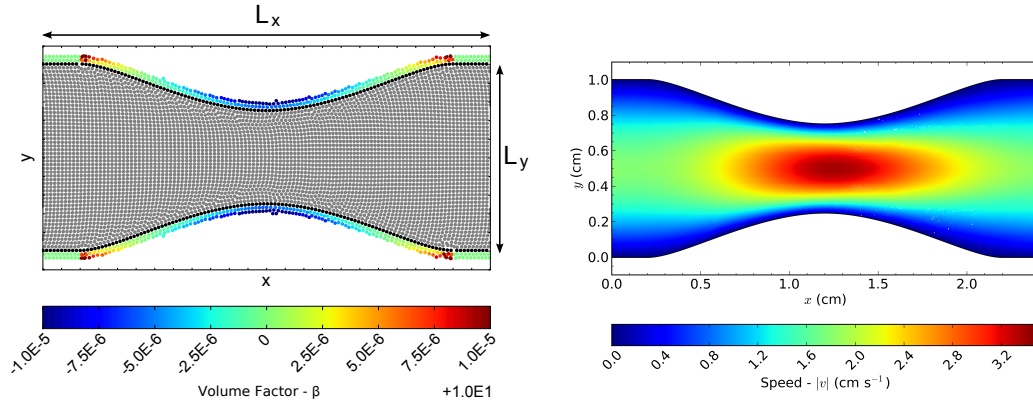


Figure 5.12: An example of a simulation using B-spline wall boundaries. In 5.12a The boundary particles are shown in black, fluid particles in grey. The reflected ghost particles are coloured according their corresponding volume-factor correction which is calculated based on the curvature of the B-spline curve. 5.12b shows the final steady-state velocity magnitude.

used as boundaries in SPH. It is reasonable to expect that the distances between neighbouring control points be similar. By taking a relatively fine grid in (t, u) this would allow us to place points onto the surface with a good coverage. These points could then be used as in 2D to provide initial guesses for t and u when locating closest points *but* they could not be used as boundary particles as their spacing would, in general, be uneven.

Another possible method for placing particles onto the grid in 3D is to place particles along the edges as for the 2D case above and keep these fixed. Then place particles on the interior of the surface approximately and use an iterative optimization technique to update their positions to evenly spread the particles. For example using a distance based potential between the particles.

5.3 Examples

5.3.1 Throttled Pipe Flow

As a simple example of using B-spline curves we show a variant of the plane Poiseuille flow in which fluid is driven through a periodic pipe under a pressure gradient, but where the pipe has a constriction. Figure 5.12 shows the simulation setup at moderate resolution with fluid particles shown in grey, boundary particles in black and the reflected particles coloured by their volume-factor correction. The size of the system is $L_x = 0.024$ m, $L_y = 0.01$ m and at the narrowest point of the constriction the width is $L_y/2$, the pressure gradient applied is 50 Pa m^{-1} and the viscosity μ is chosen to be 0.01 Pa s , giving a Reynold's number of $Re \approx 30$. The final velocity magnitude is shown in Figure 5.12b. One can see that the velocity goes to zero at the boundaries and increases in the constriction as it should. Defining the half-width

of the channel to be $R(x)$ with $R(0) = L_y/2$ we have the flux

$$Q(x) = \int_{-R(x)}^{+R(x)} v_x(x, y) dy. \quad (5.33)$$

The flux $Q(x)$ must be constant for all x due to conservation of mass. If we assume that at each x in the system the velocity profile is quadratic, in line with standard Poiseuille flow, then we can relate the velocity at the each x to the velocity at $x = 0$ via equation (5.33);

$$v_x(x, y) = \frac{R(0)}{R(x)} v_x\left(0, y \frac{R(0)}{R(x)}\right) \quad (5.34)$$

This is only an approximation since in reality we must have non-zero y components to the velocity field. At the y -midline¹, however, we would expect v_y to be zero due to symmetry. The SPH solution was compared to equation (5.34) (grey dotted line) and a validation solution (black solid line) obtained using the Fluidity package [2] which uses the finite element method. Figure 5.13b shows the profile of v_x along the line $x = L_x/2$. It matches the shape of a parabola reasonably well but there is a difference compared to both the SPH solution and the FEM validation case. Similarly figure 5.13a shows how v_x varies along the line $y = 0$ we can see it matches the shape of $v_x(x, 0)$ given by Fluidity but somewhat underestimates the magnitude of the velocity. Neither numerical solution is especially close to the form given by equation (5.34).

In this example, as is usual, c_0 was set to 10 times the maximum velocity seen in the system, but it was found that the numerical speed of sound chosen for the SPH simulation had a large effect on the shape of the solution obtained. Higher c_0 had a velocity profile which closely matched the validation case but was generally lower magnitude, whereas lower c_0 had higher magnitudes but the profile shapes did not match. Therefore it seems likely that the discrepancy of 3.5% is an artefact of the weakly compressible SPH method. The SPH and FEM solutions agree to 5 significant figures for the usual plane Poiseuille case, suggesting that the slight mismatch for this test case is caused by particle disorder. In the plane Poiseuille case the laminae of particles flow past one and other neatly; the particle positioning is highly ordered. Whereas here the laminae cannot flow through the constriction without becoming disordered, purely because fewer laminae fit across the width of the constriction. This would perhaps be remedied by the use of a truly incompressible formulation.

Despite this slight mismatch between the SPH solution and the FEM solution, the volume correction factor was found to improve the solution quality. For a simulation with 13,000 SPH particles, after a time $T = 2.5$ s, the error (relative to the FEM reference) was found to be 3.8% with volume scaling and 4.9% without.

5.3.2 Interface Forces

In this example we show a droplet of fluid spreading along a boundary due to surface forces, which are included by the continuum surface force model [18, 49]. The two fluids have a density

¹The line defined by $y = 0$

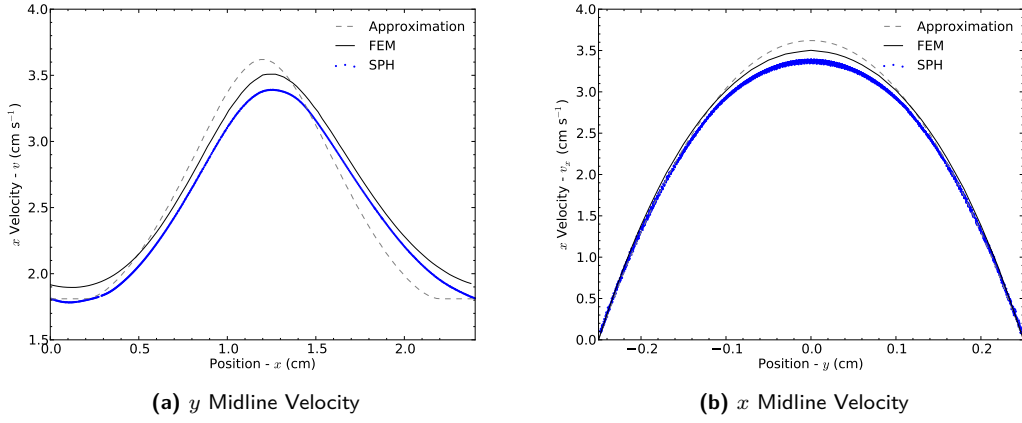


Figure 5.13: Velocity profiles for the x and y midlines, that is lines of constant y and x respectively. It should be noted that the discrepancy in Figure 5.13a is exaggerated by the limits of the y -axis, starting, as it does, from 1.5 not 0.

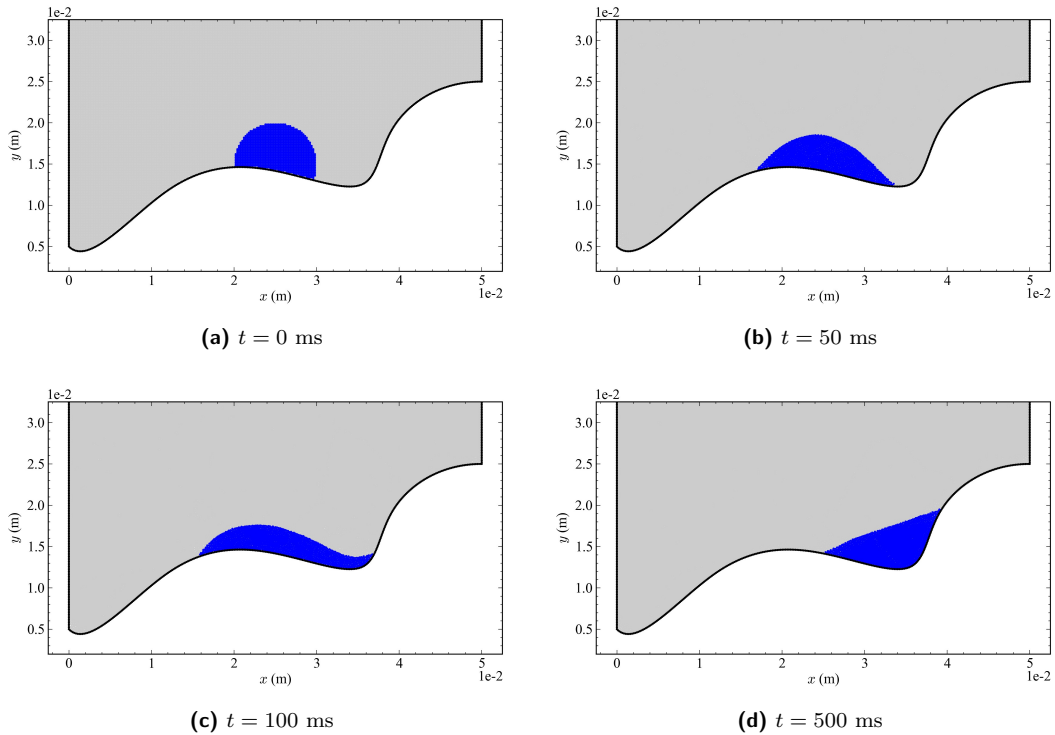


Figure 5.14: A multi-phase simulation of two fluids with contact angle of 30° and $\rho_1/\rho_2 = 10$ with boundary specified by a NURBS object.

ratio of 10, with the droplet being heavier, and equal viscosities of 0.001 Pa s. There is no gravity in this simulation. The droplet is partially wetting with a contact angle $\theta = 30^\circ$. We can see that the capillary forces at the NURBS boundary pull the droplet from its initial configuration and it becomes held in an indentation with the correct contact angle. This demonstrates that as well as shear-forces in single phase flows the NURBS boundaries can handle additional body forces.

5.4 Boundary Integral Method

In the previous sections we have examined the handling of wall boundaries using reflected ghost particles and introduced both a new correction factor for curved boundaries and a novel way of handling arbitrary curved shapes using NURBS. The next half of this chapter will discuss a new method for handling wall boundaries without the need for any ghost particles. This method, which we here call the ‘boundary integral method’, has been introduced in the literature by Kulasegaram et al and Ferrand et al [37, 63]. In this thesis we introduce a novel derivation for the gradient combining this technique with the zero order corrected kernel [equation \(5.36\)](#), a new approximation for the boundary terms which utilizes the boundary particles and, lastly, apply these boundary conditions to multiphase simulations. An issue which, to the best of our knowledge, has not been discussed in the literature.

5.4.1 Kernel Inconsistency

As discussed in [section 3.1.2](#) particle inconsistency occurs when a non regular positioning of particles leads to the constraints on the smoothing kernel ([table 3.1](#)) for the discretized SPH sums not being satisfied. This can be corrected up to zeroth order using the form of the SPH interpolant given by [equation \(3.16\)](#). A related but different issue occurs near the boundaries of the domain for the continuum versions of the smoothing operations.

If $\mathbf{x}$ is within $2h$ of $\partial\Omega$ then the kernel support $\Omega_{\mathbf{x}}$ will lie partly outside of the domain. This means that the kernel no longer integrates to unity over this region ($\Omega \cap \Omega_{\mathbf{x}}$), again breaking the required constraints. This is one aspect of what is referred to above as kernel boundary deficiency; though that is a combination of both particle and kernel inconsistency.

Analogously to [equation \(3.16\)](#) we can correct this error on the smoothing kernel by using the kernel support correction factor γ . γ represents how much of the weight of the smoothing kernel is not missing from the integration domain, it is defined to be

$$\gamma(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} W(|\mathbf{x} - \mathbf{y}|) d\mathbf{y} \quad (5.35)$$

Since we require that the integral of the weighting function (smoothing kernel) is always equal to one, we multiply our SPH interpolants by a factor $\frac{1}{\gamma}$ thereby ensuring the integral is normalized, even near the domain edges.

As well as renormalizing the kernel we must explicitly account for the surface integral terms

arising in the derivation of [equation \(3.7\)](#) in [section 3.1.1](#). Our updated SPH interpolation function is

$$\langle \phi \rangle(\mathbf{x}_i) = \frac{1}{V_i \gamma_i} \int_{\Omega_{\mathbf{x}_i}} \chi_i(|\mathbf{x}_i - \mathbf{y}|) d\mathbf{y}. \quad (5.36)$$

As in [section 3.1.1](#) the gradient $\nabla_{\mathbf{x}_i} \phi$ is substituted in to [equation \(5.36\)](#), the integral split using the chain rule, and the first integral term is then converted to a surface integral term by using the Gaussian integral theorem. By convention we take the normals of our domain boundaries $\hat{\mathbf{n}}$ to be pointing inwards to the domain, thus the resulting term gains a minus sign since the theorem is valid for *outward* pointing normals, leading to

$$\langle \nabla_{\mathbf{x}} \phi \rangle(\mathbf{x}) = -\frac{1}{V_i \gamma_i} \oint_{\partial \Omega_{\mathbf{x}}} \chi_i(|\mathbf{x} - \mathbf{y}|) \phi(\mathbf{y}) \hat{\mathbf{n}} dS - \frac{1}{V_i \gamma_i} \int_{\Omega_{\mathbf{x}}} \nabla_{\mathbf{y}} \chi_i(|\mathbf{x} - \mathbf{y}|) \phi(\mathbf{y}) d\mathbf{y}. \quad (5.37)$$

The surface integral is over the closed surface of the region $\Omega \cup \Omega_{\mathbf{x}}$ however the integrand is exactly zero at a radius of $2h$ thus the only contribution comes from the integral along the domain boundaries $\partial \Omega \cap B_{2h}(\mathbf{x})$. Note the difference here is that we have explicitly accounted for the surface integral term, which away from the domain boundaries is zero. In order to calculate an approximation for this integral we will make use of boundary particles².

Using the kernel gradient [equation \(A.62\)](#) in the volume integral term and approximating the surface integral term as a sum over boundary particles it is possible to show the SPH gradient estimate is

$$\nabla A_i \approx \frac{1}{V_i \gamma_i} \sum_{j \in \mathcal{F}, \mathcal{B}} \left(\frac{A_i \gamma_i^2}{\sigma_i^2} + \frac{A_j \gamma_j^2}{\sigma_j^2} \right) \nabla W_{ij} - \frac{1}{V_i \gamma_i} \sum_{b \in \mathcal{B}} \left(\frac{A_i \gamma_i}{\sigma_i} + \frac{A_b \gamma_b}{\sigma_b} \right) W_{ib} \hat{\mathbf{n}}_b dS \quad (5.38)$$

The second term in this equation (the sum over boundary particles) represents the correction to our gradient estimator which arises from being near the simulation domain boundary. From our derivation the resulting surface terms contain γ/σ as opposed to in [\[37\]](#), where even the surface terms contain $(\gamma/\sigma)^2$, however both choices produce functioning wall boundaries.

5.4.2 Calculating γ

It is clear from the definition of γ that

$$\nabla \gamma(\mathbf{x}) = \int_{\partial \Omega_{\mathbf{x}}} W(|\mathbf{x} - \mathbf{x}_S|) \hat{\mathbf{n}}_S dS. \quad (5.39)$$

This can be approximated as a sum over nearby boundary particles;

$$\nabla \gamma_i \approx \sum_b W_{ib} \hat{\mathbf{n}}_b \Delta S \quad (5.40)$$

To calculate γ_i through the course of a simulation it is possible to integrate it through time using the relation [\[37\]](#)

$$\frac{d\gamma_i}{dt} = \nabla \gamma_i \cdot \mathbf{u}_i. \quad (5.41)$$

²See [section 5.4.3](#).

This relies, however, on knowing the value of γ at time $t = 0$. In Ferrand 2012 [37] they use a ‘fictitious’ time integral at the zeroth time step to initialize γ . They move the particle to a part of the domain $> 2h$ away from any boundaries by moving in a straight line along the direction $\nabla\gamma_i$. Far from any walls γ_i is one and then we can use [equation \(5.41\)](#) to integrate along the path back to the particle’s original position, thereby giving us its value of γ .

This approach works well for systems with few boundaries and large regions which are far from the walls. For the complicated geometries examined in this thesis this approach was found not to work. For this reason we initialize γ by using a direct numerical integral rather than a numerical path integral.

If the boundary consists of line segments then drawing rays from the point x to the ends of the line segments forms a series of triangles. On each of these triangles the integral of W contributes to the total value of γ . For a given radius r we can write

$$\left. \frac{d\gamma}{d\theta} \right|_r = \int_0^r s W(s) ds := G(r), \quad (5.42)$$

which can be easily evaluated and is just a polynomial in r . Then, as we move along a line segment s constituting part of the boundary, the total contribution from the triangle formed by the end points of s with $\mathbf{x}$ will be given by

$$\gamma_s = \int_{\theta_1}^{\theta_2} G(r(\theta)) d\theta \quad (5.43)$$

Despite the fact that $G(r)$ is just a polynomial in r the form of $r(\theta)$ makes it unfeasible to solve this integral analytically. Given we are working with a triangular region it was hoped rewriting the integral using barycentric coordinates would help. If the two end points of s are given by $\mathbf{a}$ and $\mathbf{b}$ respectively then we have

$$\gamma_s = \int_0^1 \int_0^{1-\alpha} W\left(\sqrt{(\alpha\mathbf{a} + \beta\mathbf{b}) \cdot (\alpha\mathbf{a} + \beta\mathbf{b})}\right) d\beta d\alpha. \quad (5.44)$$

However, despite looking simpler this integral remains difficult to solve³.

The reason this integral turns out not to be practically solvable is because the distance is calculated as a square-root of a quadratic polynomial in α and β , but crucially this quadratic is *not* a perfect square and so the square-root is no longer a polynomial. Substituting this into W which, in this thesis, is a fifth order polynomial compounds the problem. It is also for this reason that $r(\theta)$ has a form, which renders the integral [equation \(5.43\)](#) unsolvable.

It is worth noting that if our smoothing kernel consisted of only even powers of r then this integral, while containing many terms, would actually be easily solvable because the even powers would still be integer powers after square-rooting. However it appears difficult to construct a smoothing kernel which satisfies the necessary conditions *and* is sufficiently peaked using only even powers of r without using many, many terms.

³The computer algebra system Mathematica running for several hours was unable to find a solution

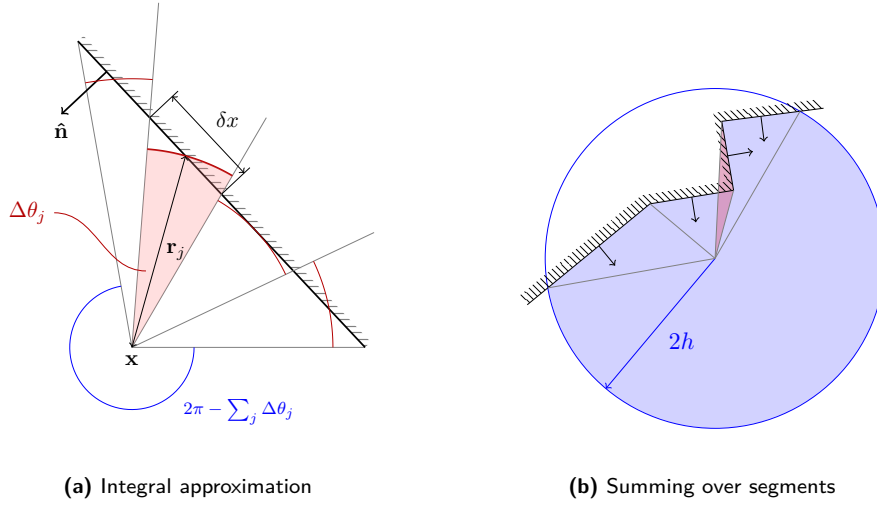


Figure 5.15: The value of γ is calculated by approximating the polar integral over the region $\Omega_{\mathbf{x}}$.

Because we cannot in general calculate the value of this integral analytically we approximate it numerically. Each boundary segment s is sub-divided into segments of length δx (not necessarily the same as the particle spacing Δx) where the end points of each sub-segment j are given by $\mathbf{x}_{j-1}$ and $\mathbf{x}_{j+1}$ and the midpoint is given by $\mathbf{x}_j$. Write $\mathbf{r}_j = \mathbf{x}_j - \mathbf{x}$, the vector from our particle to the segment end- and midpoints. Then the angle spanned by the sub-segment is

$$\Delta\theta_j = \arccos\left(\frac{\mathbf{r}_{j-1} \cdot \mathbf{r}_{j+1}}{r_{j-1}r_{j+1}}\right). \quad (5.45)$$

This small segment with length δx forms a triangle with the point $\mathbf{x}$. Because δx is small, the integral [equation \(5.43\)](#) can be approximated by assuming the segment is at a constant radius giving

$$\gamma_s \approx \Delta\theta G(r_j). \quad (5.46)$$

The contribution from this small segment however should only be added if the segment is facing the point $\mathbf{x}$, otherwise it should actually be subtracted (c.f. [Figure 5.15b](#)). To check if the contribution should be added or subtracted one can simply check the dot product of the wall normal vector $\hat{\mathbf{n}}$ with $\mathbf{r}_j$; $(\hat{\mathbf{n}} \cdot \mathbf{r}_j) < 0$ implies that the segment is facing us and the contribution must be added not subtracted. Therefore the total contribution from the subsegments is

$$\gamma_s = - \sum_j \text{sgn}(\hat{\mathbf{n}} \cdot \mathbf{r}_j) \Delta\theta_j G(r_j). \quad (5.47)$$

Together the triangles cover an angle given by

$$\Theta = - \sum_j \text{sgn}(\hat{\mathbf{n}} \cdot \mathbf{r}_j) \Delta\theta_j. \quad (5.48)$$

which means that the rest of the support extends to $2h$ and its contribution to γ must be

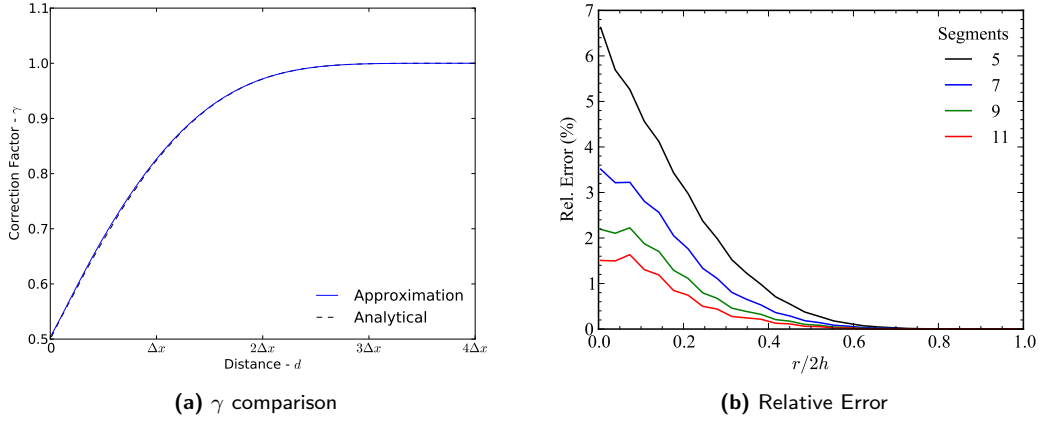


Figure 5.16: (left) Comparison of the approximation for γ given by equation (5.49) with the value of γ computed using equation (5.41). (right) For very small distances from the wall we can see large errors when we discretise the boundary using only a few segments. This error rapidly decreases as we get further away and as we take more segments. Based on this it seems necessary to take > 10 segments when calculating correction factor.

$1 - \Theta/2\pi$, see figure 5.15b. Summing these two contributions gives

$$\gamma(\mathbf{x}) = \left(1 - \frac{\Theta}{2\pi}\right) - \sum_j \text{sgn}(\hat{\mathbf{n}} \cdot \mathbf{r}_j) \Delta\theta_j G(r_j). \quad (5.49)$$

This approximation is reasonably accurate but it will become progressively more inaccurate the closer $\mathbf{x}$ is to a wall since the angle subtended by segments near $\mathbf{x}$ will become larger. Shown in figure 5.16a is the approximation for γ given by equation (5.49) compared to the value for γ computed using the line integral of equation (5.41) for a flat wall. The subsegments were given a length Δx equal to that of the particle spacing and the smoothing factor is $s = 2$. For the example in figure 5.16a the error is very small ($< 1\%$) for all distances but depending on whether our SPH particle has a displacement parallel to the wall we can see errors in the integration for very small distances of up to 10%. Figure 5.16b shows the worst case error for a small distance away from the wall ($0.01\Delta x$) over all transverse displacements relative to the boundary particles used. It is clear that to obtain a consistently usable approximation one must take the discretization length to be ~ 10 times smaller than Δx .

5.4.3 Boundary Particles

The boundary particles are placed along the boundary with a spacing of $\Delta x B_f$ thus the area⁴ that each boundary particle represents is $dS = (\Delta x B_f)^{d-1}$. Each boundary particle lies partly inside the domain and partly outside. For a flat wall one can imagine that they lie half in the domain in and half out. For a curved boundary or at a corner this factor is different from 1/2 and depends on either the curvature κ or the angle of the corner θ . The ‘amount’ of the boundary particle b inside the domain is $K_b = V_b/\Delta x^d$. Whilst subtly different, we can see that

⁴Length in two dimensions.

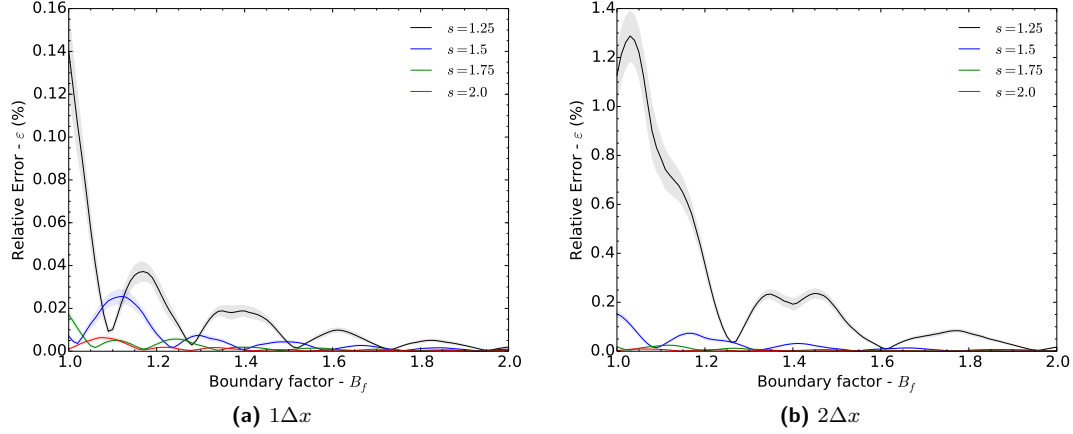


Figure 5.17: The relative error in approximating the surface integral term for a range of boundary factors B_f and smoothing length factors s . The two graphs show the error for a particle Δx and $2\Delta x$ away from a flat boundary respectively.

$$K_b \approx \gamma_b.$$

Because they straddle the domain boundary, the boundary particles contribute both to the volume integral and surface integral approximation of [equation \(5.37\)](#). For the volume integral part we weight their contribution by $K_b B_f$, which ensures the correct normalization of the kernel. To aid clarity these factors will often not be explicitly included in the SPH equations as written.

Because the smoothing kernel is highly peaked around $r = 0$ one might worry about the ability of the surface integral approximation to accurately reproduce the correct value. To check whether this was likely to be a problem we calculated the approximate value of the surface integral for a constant function and compared it to an answer obtained using an adaptive quadrature routine⁵. Figures 5.17a and 5.17b show the relative error of the discretization for a particle Δx and $2\Delta x$ away from a flat boundary respectively. Note that in [figure 5.17b](#) the line for smoothing factor $s = 1.0$ is not shown because at a distance of $2\Delta x$ there is no overlap between the support of W and the boundary. As can be seen in [figure 5.17a](#) even for low s and low B_f the relative error between the simple approximation and more accurate computation is very small, being less than one quarter of one percent. For the particles at a distance of $2\Delta x$ the error is larger due to there being fewer boundary particles in the support of W but, not only is this error still small, being of order 1%, but the overall contribution is much lower due to the small value of W away from $r = 0$.

This suggests one can take $B_f = 1$ without introducing unnecessarily large errors into the simulation. If, however, one wished to increase the resolution of particles on the boundary, relative to the fluid resolution, the increase in computation needed goes as $O(B_f^{d-1})$. Since the boundaries usually constitute a small part⁶ of the simulation domain this is very unlikely to be a performance bottleneck.

⁵scipy.integrate.quad which uses Fortran QUADPACK.

⁶In terms of the number of boundary particles to fluid particles.

5.4.4 Particle Spacing Factor

The factor σ which corrects for particle inconsistency must also be calculated carefully near the boundaries. It is calculated purely as a ‘volume integral’ type term given by

$$\sigma_i = \sum_{j \in \mathcal{F}} W_{ij} + \sum_{b \in \mathcal{B}} W_{ib} K_b B_f. \quad (5.50)$$

The extra factors in the boundary particle sum account for the fact the boundary particles are lying partly outside the domain.

5.4.5 Physical Boundary Conditions

Once σ has been calculated using [equation \(5.50\)](#), and the density has been calculated from σ , we can use [equation \(5.38\)](#) directly, to calculate the pressure gradient

$$\begin{aligned} -\nabla P_i = & -\frac{1}{\gamma_i V_i} \sum_j \left(\frac{P_i \gamma_i^2}{\sigma_i^2} + \frac{P_j \gamma_j^2}{\sigma_j^2} \right) \nabla W_{ij} \\ & + \frac{1}{\gamma_i V_i} \sum_b \left(\frac{P_i \gamma_i}{\sigma_i} + \frac{P_b \gamma_b}{\sigma_b} \right) \hat{\mathbf{n}}_b W_{ib} dS. \end{aligned} \quad (5.51)$$

For the viscous stress a little more care must be taken. The fluid-fluid (volume integral) contribution to the viscous stress is obtained by combining the fluid-fluid term from [equation \(5.38\)](#) with the usual SPH estimator for the viscous force [equation \(3.32\)](#). This yields

$$(\nabla^2 \mathbf{u})_i^{\text{vol.}} = \frac{\mu}{\gamma_i V_i} \sum_j \left(\frac{\gamma_i^2}{\sigma_i^2} + \frac{\gamma_j^2}{\sigma_j^2} \right) \frac{\mathbf{u}_{ij}}{x_{ij}} \frac{\partial W_{ij}}{\partial r}, \quad (5.52)$$

which is the γ -corrected volume integral part of the gradient applied to $\mu \nabla^2 \mathbf{u}$. When the particles i and j are both $> 2h$ from a domain boundary this recovers exactly [equation \(3.32\)](#).

In the continuous version of the surface integrals the fluid-boundary terms manifest themselves with the form $\hat{\mathbf{n}}_b \cdot \nabla \mathbf{u}$. That is, they appear as the directional derivative of the velocity perpendicular to the wall. We can perform a similar approximation for $\hat{\mathbf{n}}_b \cdot \nabla \mathbf{u}$ as for the volume term. In the volume term we approximate the directional derivative $\hat{\mathbf{e}}_{ij} \cdot \nabla \mathbf{u}$ with

$$\hat{\mathbf{e}}_{ij} \cdot \nabla \mathbf{u} \approx \frac{\mathbf{u}_{ij}}{x_{ij}}.$$

In order to make a similar approximation for the boundary-fluid pair, we assume that the velocity is changing only in the direction perpendicular to the wall. Our fluid particle i , say, is a distance x_{ib} away from our boundary particle b but a distance $(\mathbf{x}_{ij} \cdot \hat{\mathbf{n}}_b)$ away from the wall. Therefore if all of the change in velocity has come from moving away from the wall (as opposed to along the wall) then the directional derivative is given by

$$\hat{\mathbf{n}}_b \cdot \nabla \mathbf{u} \approx \frac{\mathbf{u}_{ij}}{(\mathbf{x}_{ij} \cdot \hat{\mathbf{n}}_b)}. \quad (5.53)$$

This approximation for the directional derivative can then be substituted into the boundary contribution term in [equation \(5.38\)](#).

Combining these gives the total acceleration due to viscosity (including boundary contribution);

$$\begin{aligned} \nabla^2 \mathbf{u}_i = & \frac{\mu}{\gamma_i V_i} \sum_j \left(\frac{\gamma_i^2}{\sigma_i^2} + \frac{\gamma_j^2}{\sigma_j^2} \right) \frac{\mathbf{u}_{ij}}{x_{ij}} \frac{\partial W_{ij}}{\partial r} \\ & - \frac{\mu}{\gamma_i V_i} \sum_b \left(\frac{\gamma_i}{\sigma_i} + \frac{\gamma_b}{\sigma_b} \right) \frac{\mathbf{u}_{ij}}{(\mathbf{x}_{ij} \cdot \hat{\mathbf{n}}_b)} W_{ib} dS. \end{aligned} \quad (5.54)$$

This approximation for the viscosity is valid because for ideal no slip boundary conditions we would expect that near the walls we have

$$\lim_{r \rightarrow 0} \frac{\|\mathbf{u}(r)\|}{\|(\hat{\mathbf{n}} \cdot \nabla) \mathbf{u}\| r} = 1$$

where r is the (perpendicular) distance from the wall. This equation says that at small distances the velocity near the boundary varies linearly.

5.5 Results

5.5.1 Hydrostatic Case

As a first test of the new boundary conditions we examine a very simple hydrostatic case in which water fills a circular container of radius 4.5 cm. The BIM implicitly handles the curvature of the circular container wall unlike for reflected boundary conditions where the curvature must be explicitly accounted for as detailed in [section 5.1](#).

It can be seen from [figure 5.18](#) that the boundaries function well in this case. The fluid particles arrange themselves correctly at the boundary ([figure 5.18a](#)) and there are no unwanted velocities arising. Further, examining the plot of γ for particles which are near the boundary we can see that the time stepping scheme [equation \(5.41\)](#) works as expected and that the particles' values of γ all lie on the expected theoretical curve shown in grey.

The integrated boundaries actually perform slightly better than the reflected boundaries for this test case. Due to the initial positioning of the particles being on a cubic grid there is some rearrangement near the boundaries to even out the particle distribution. This causes some non-zero velocities initially. For the BIM simulation these velocities are consistently lower than for the reflected boundary conditions, see [figure 5.19](#). While the difference might not look large from the graph, by $t = 1$ s the mean velocity in the BIM simulation is approximately 40% of that of the reflection simulation. Given that the analytical solution is a uniform zero velocity field this means that the error in the SPH solution is actually better for the BIM than for reflection.

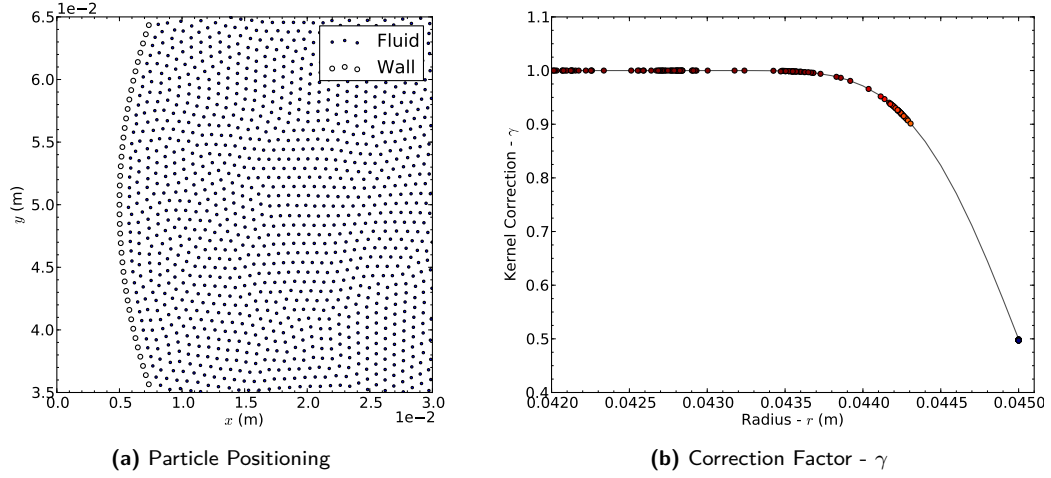


Figure 5.18: (a) The particle positions for the BIM simulation show that the particle nicely arrange themselves at the boundaries and there is no penetration across the boundary. (b) The kernel correction factor γ plotted against radius is compared to the theoretical curve (grey).

5.5.2 Poiseuille Flow

To test the viscosity boundary conditions we check the performance of the plane-Poiseuille flow in 2D and compare with the current reflecting boundaries of [48]. The time varying analytical solution for plane-Poiseuille flow is given by

$$u_y(x, t) = \frac{g\rho}{2\mu}(x^2 - D^2) + \sum_{n=0}^{\infty} \frac{16(-1)^n D^2 g \rho}{\mu \pi^3 (2n+1)^3} \times \cos\left(\frac{(2n+1)\pi x}{2D}\right) \exp\left(-\frac{(2n+1)^2 \pi^2 \mu t}{\rho 4d^2}\right), \quad (5.55)$$

where g is the acceleration on the fluid due to gravity (or alternatively a pressure gradient), $D = L/2$ is half the plate separation, and ρ and μ have their usual meanings, see [91]. The infinite time, steady state, solution is therefore given by

$$u_y(x) = \frac{g\rho}{2\mu}(x^2 - D^2). \quad (5.56)$$

The results from an SPH simulation using the new boundary conditions are shown in figure 5.20 showing the evolution of the velocity profile with time for both the SPH solution and the analytical value. The viscosity formulation works well, although in the long time limit the velocity overshoots by $\approx 0.5\%$ at the lowest resolution.

5.6 Gradient Correction Order

It is possible to show that the gradient correction detailed is 1st order, that is to say it can exactly estimate the gradient for linear functions. Consider the scalar field $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ defined

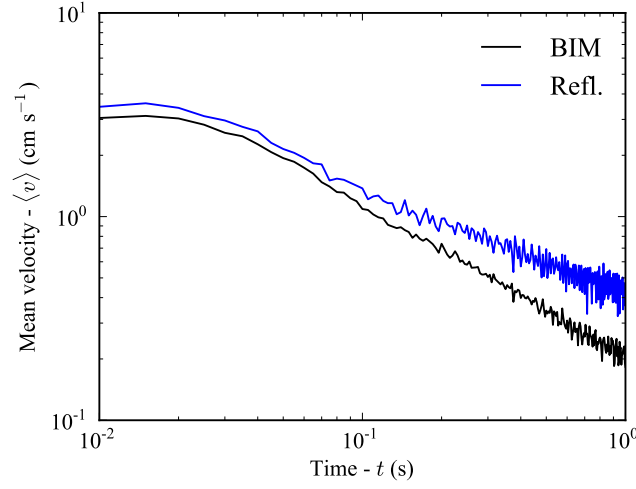


Figure 5.19: The initial rearrangement of SPH particles causes some non zero velocities. These die away quicker in the BIM simulation than the reflection simulation.

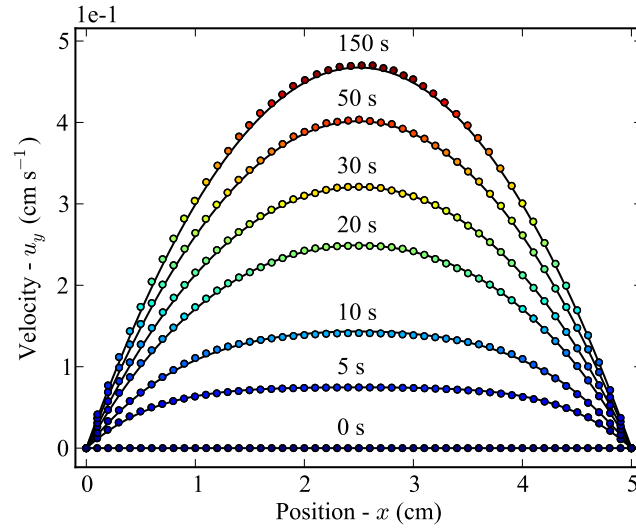


Figure 5.20: Time evolution of the velocity profile in plane-Poiseuille flow. The dots show the results of an SPH simulation with $s = 2.0$ and $\Delta x = 1$ mm, the solid lines show the analytical solutions at various times.

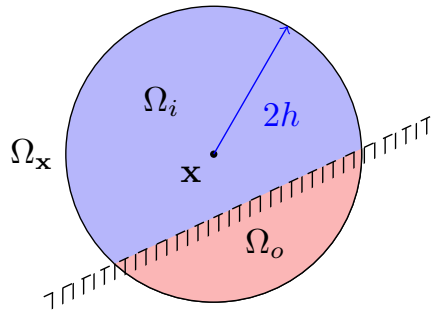


Figure 5.21: Full integration domain Ω_x with a fictitious cut dividing the domain into two subdomains Ω_i and Ω_o .

by $\phi(\mathbf{x}) = a + \mathbf{b} \cdot \mathbf{x}$ which varies linearly in each coordinate. Imagine we wish to calculate the gradient near the domain boundary so that the full kernel support $\Omega_{\mathbf{x}}$ is split into two parts; Ω_i which is interior to Ω and Ω_o which is exterior (see [figure 5.21](#)). We will compare the full integral over $\Omega_{\mathbf{x}}$ to the surface corrected integral over only Ω_i . For the correction to be at least first order we require

$$\int_{\Omega_{\mathbf{x}}} W(|\mathbf{x} - \mathbf{y}|) \nabla \phi(\mathbf{y}) d\mathbf{y} = \frac{1}{\gamma} \int_{\Omega_i} W(|\mathbf{x} - \mathbf{y}|) \nabla \phi(\mathbf{y}) d\mathbf{y}. \quad (5.57)$$

Since we know $\nabla \phi = \mathbf{b}$ this gives us

$$\mathbf{b} \gamma \int_{\Omega_{\mathbf{x}}} W d\mathbf{y} = \mathbf{b} \int_{\Omega_i} W d\mathbf{y}. \quad (5.58)$$

Canceling the $\mathbf{b}$'s

$$\underbrace{\gamma \int_{\Omega_{\mathbf{x}}} W d\mathbf{y}}_{=1} = \underbrace{\int_{\Omega_i} W d\mathbf{y}}_{=\gamma}. \quad (5.59)$$

Thus we see that the corrected gradient is indeed exactly equal to the usual estimate over $\Omega_{\mathbf{x}}$ for linear functions. Now consider the general quadratic function

$$\phi(\mathbf{x}) = a + \mathbf{b} \cdot \mathbf{x} + \mathbf{x}^T \mathbf{C} \mathbf{x}, \quad (5.60)$$

The matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$ is a symmetric matrix which defines the coefficients of the quadratic terms. The gradient of ϕ is given by

$$\nabla \phi(\mathbf{x}) = \mathbf{b} + 2\mathbf{C} \cdot \mathbf{x}. \quad (5.61)$$

Substituting into both sides of (5.57) gives

$$\frac{1}{\gamma} \left(\mathbf{b} \int_{\Omega_i} W d\mathbf{y} + 2\mathbf{C} \cdot \int_{\Omega_i} \mathbf{y} W d\mathbf{y} \right) = \mathbf{b} \int_{\Omega_{\mathbf{x}}} W d\mathbf{y} + 2\mathbf{C} \cdot \int_{\Omega_{\mathbf{x}}} \mathbf{y} W d\mathbf{y}. \quad (5.62)$$

The terms arising due to the linear part are equal (shown above) so we can subtract them from both sides and focus on the terms arising from the quadratic part of ϕ . Multiplying by γ and using the distributive property of matrix multiplication gives

$$2\mathbf{C} \cdot \left(\int_{\Omega_i} \mathbf{y} W d\mathbf{y} - \gamma \int_{\Omega_{\mathbf{x}}} \mathbf{y} W d\mathbf{y} \right) = 0. \quad (5.63)$$

Since in general $\det(\mathbf{C}) \neq 0$ the only solution to [equation \(5.63\)](#) is

$$\int_{\Omega_i} \mathbf{y} W d\mathbf{y} - \gamma \int_{\Omega_{\mathbf{x}}} \mathbf{y} W d\mathbf{y} = 0, \quad (5.64)$$

$$\Rightarrow \int_{\Omega_i} \mathbf{y} W d\mathbf{y} = \gamma \int_{\Omega_{\mathbf{x}}} \mathbf{y} W d\mathbf{y}. \quad (5.65)$$

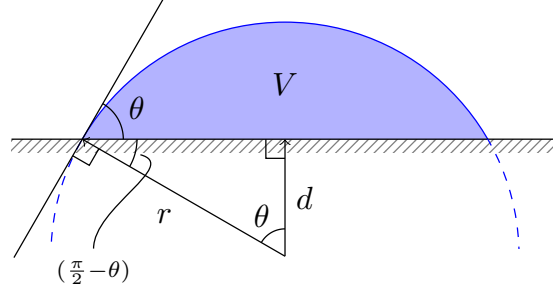


Figure 5.22: In 2D a droplet against a wall, without gravity, will form a circular segment. The radius r and centre of the circle $(0, -d)$ depend on the the volume V and the contact angle θ .

Using a change of variable $\mathbf{y}' = \mathbf{x} - \mathbf{y}$ in [equation \(5.65\)](#) leads to

$$\int_{\Omega_i} \mathbf{y}' W(|\mathbf{y}'|) d\mathbf{y}' = \gamma \int_{\Omega_x} \mathbf{y}' W(|\mathbf{y}'|) d\mathbf{y}', \quad (5.66)$$

the right hand side of which is equal to zero (due to zero mean constraint on W) giving

$$\int_{\Omega_i} \mathbf{y}' W(|\mathbf{y}'|) d\mathbf{y}' = 0. \quad (5.67)$$

In general [equation \(5.65\)](#) is not true, thus the corrected interpolation given by [equation \(5.36\)](#) is incapable of correctly recreating gradients of second order (or higher) functions with missing kernel support.

5.7 Contact Angle and Surface Forces

The boundary conditions as detailed up to now work well for pressure and viscosity because they both vary approximately linearly across the region $\Omega_{\mathbf{x}}$ and so the corrected gradient works well to estimate the gradients. However, things are more complicated when calculating the forces due to surface tension.

5.7.1 Test Case

To test the correctness of the contact angle formulations we will simulate a droplet against a flat wall with a specified contact angle θ in two dimensions with no gravity. The equilibrium shape for this droplet is an arc of a circle meeting the wall with an angle θ . Given the droplet volume V one can calculate the expected shape of the interface by calculating the radius and centre of the circle forming the corresponding circular segment. The geometry is shown in [figure 5.22](#).

By simple trigonometry one can obtain r and d in terms of the knowns, V and θ ;

$$r = \sqrt{\frac{V}{\theta - \cos(\theta) \sin(\theta)}}, \quad (5.68)$$

and

$$d = \begin{cases} r \cos(\theta) & \text{if } \theta \leq \pi/2 \\ r \sin(\pi/2 - \theta) & \text{if } \theta > \pi/2 \end{cases}. \quad (5.69)$$

Using this we can check visually whether the simulated droplet takes on the correct shape. To quantify the shape of the droplet we can measure the width and height it assumes. These are related to the radius and centre of the circular segment;

$$w = 2r(\theta) \sin(\theta), \quad (5.70)$$

is the width and

$$H = r(\theta) (1 - \cos(\theta)), \quad (5.71)$$

is the height. Since the interface in SPH is not explicitly tracked we cannot compare the shape of the droplet but we can measure the width and height based on the positions of the SPH particles.

5.7.2 Colour Gradients

As discussed in [chapter 3](#), to calculate the forces on our fluids due to surface tension we need to know the gradients of colour, both for the prefactor⁷ and for calculating $\hat{\mathbf{n}}$. Away from any boundaries the colour gradients can be calculated as normal. Near boundaries care must be taken but the colour gradients can still be exactly calculated. The smoothed gradient of C^k over $\Omega_{\mathbf{x}}$ as given by the normal SPH estimate is

$$\langle \nabla C^k \rangle(\mathbf{x}) = \frac{1}{V_i} \int_{\Omega_{\mathbf{x}}} C^k(\mathbf{y}) \nabla \chi_i(|\mathbf{x} - \mathbf{y}|) d\mathbf{y}.$$

This integral can be split into the sum of integrals over the interior and exterior ‘domains’ - Ω_i and Ω_o ;

$$\langle \nabla C^k \rangle(\mathbf{x}) = \frac{1}{V_i} \int_{\Omega_i} C^k(\mathbf{y}) \nabla \chi_i(|\mathbf{x} - \mathbf{y}|) d\mathbf{y} + \frac{1}{V_i} \int_{\Omega_o} C^k(\mathbf{y}) \nabla \chi_i(|\mathbf{x} - \mathbf{y}|) d\mathbf{y}. \quad (5.72)$$

Now imagining that Ω_o represents the region outside our simulation domain Ω we know that C^k is constant over this domain (it is the wall colour). If k is the same as the wall colour this constant is 1, thus the second integral in (5.72) can be converted simply to a surface integral giving

$$\nabla C^k = \frac{1}{V_i} \int_{\Omega_i} C^k \nabla \chi_i d\mathbf{y} + \frac{1}{V_i} \int_{\partial\Omega_i} \chi_i \hat{\mathbf{n}}_s dS, \quad (5.73)$$

⁷Used to turn the interface force into a body force.

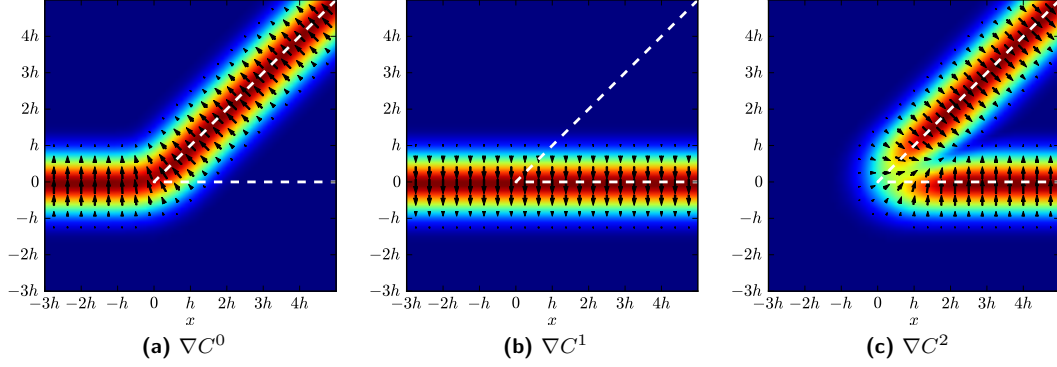


Figure 5.23: The colour gradient for three phases with a contact angle of $\theta = 45^\circ$.

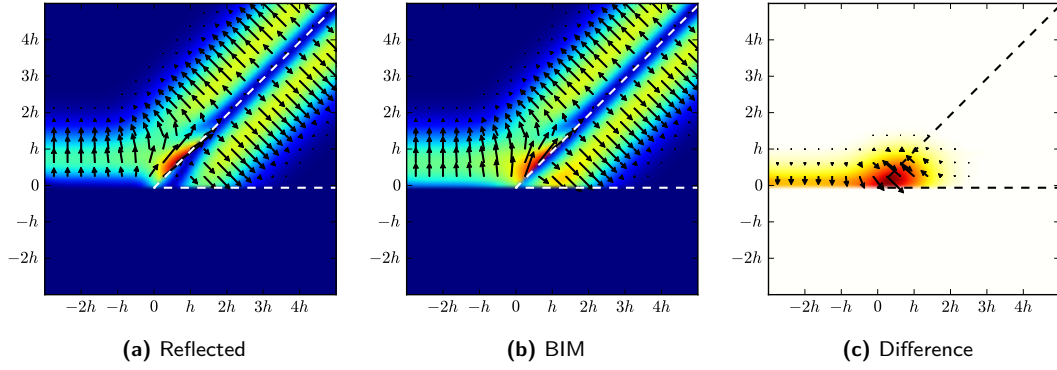


Figure 5.24: The surface tension force at equilibrium shows the inability of the SA boundaries to capture the forces correctly, as well as features such as the spurious velocities. The forces normal to the interface are balanced by the Laplace pressure.

where we have used the fact that the surface integrals over $\partial\Omega_i$ and $\partial\Omega_o$ are necessarily equal and opposite as well as the fact the normals $\hat{\mathbf{n}}$ are defined to point into the domain. Thus this represents the exact colour gradient for the wall colour k . If k is some other colour then the surface integral contribution disappears (since C^k would be zero for all $\mathbf{x} \in \Omega_o$) and the volume term is enough to calculate ∇C^k exactly. Note that this is a different form than was used for $-\nabla P$ and $\mu \nabla^2 \mathbf{v}$, but we are still able to exactly calculate the colour gradients near the boundaries.

5.7.3 Continuum Surface Force

The surface tension forces for a continuum example were calculated using the usual integral and the corrected form. The example is at its equilibrium contact angle of 45° . We can see, [figure 5.24a](#), that at the equilibrium contact angle the usual integral gives a force inwards to the phases (normal to the interface) which is balanced by the Laplace pressure. Also visible are the forces tangential to the interface which are the cause of the spurious velocity currents⁸. In

⁸See [section 6.1](#)

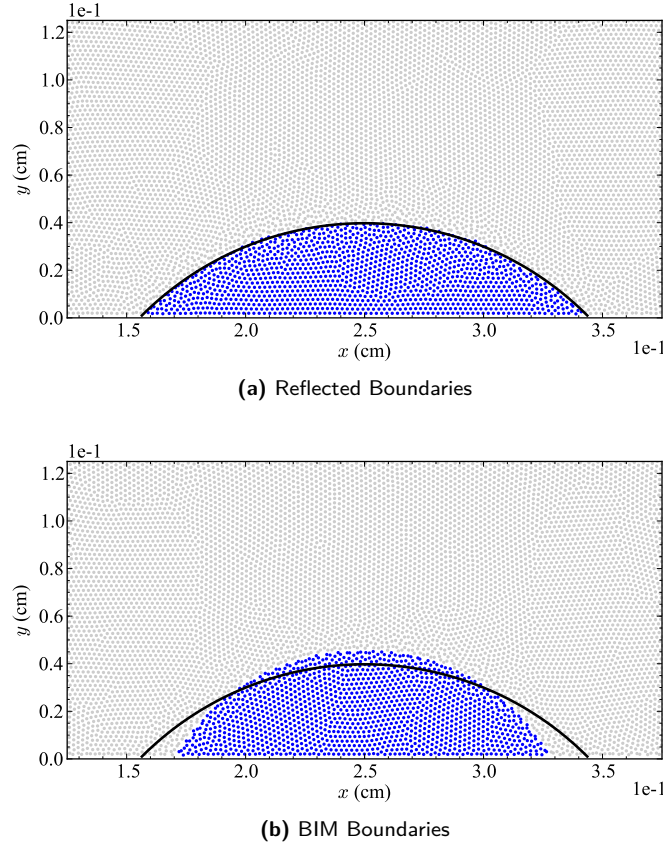


Figure 5.25: The usual reflected ghost particle boundaries are able to correctly capture the specified contact angle (figure 5.25a) unlike the boundary integral method boundaries (figure 5.25b) which cannot produce the correct contact angle. The two simulations had exactly the same parameter values.

figure 5.24b we see the same force calculated using the BIM. There is a slight discrepancy at the contact point itself. Moreover, for the non-wetting phase the inward force due to interaction with the substrate is mis-estimated.

This was confirmed by a test case which fails to behave as it physically should. A droplet of fluid with volume 0.5 cm^3 was simulated sitting against a substrate for which a range of equilibrium contact angles were specified. There was no gravity meaning the theoretical droplet shape is given by a circular arc as discussed above. For the usual reflected ghost boundary conditions the CSF model displays the correct dynamics and the droplet assumes the correct shape. Using the corrected gradient of the boundary integral method, however, does not correctly capture the forces resulting from the CSF model. A comparison of the final state of the droplet for a contact angle of 60° is shown in figure 5.25. The miscalculation of the surface forces is more pronounced the further θ is from 90° ; at $\theta = 30^\circ$ the relative error of w is 29.5%, at $\theta = 150^\circ$ it is 61.3% but at $\theta = 90^\circ$ it is just 1.2%.

The gradients of colour needed for most surface tension models in SPH are effectively convolutions of the smoothing kernel gradient ∇W with the colour function. Since the colour function

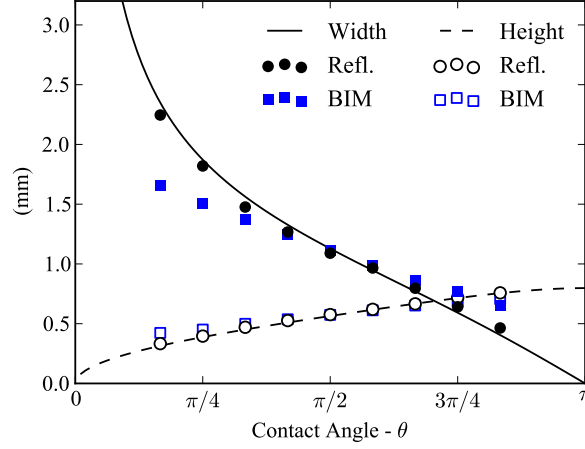


Figure 5.26: The analytical droplet width and height for varying θ are compared to the results from a simulation with reflected BCs and BIM BCs. The reflected particle method droplet forms the correct shape whereas the BIM fails to capture the correct shape.

is a step function this means $\nabla W * C^k$ will be of the same order as the smoothing kernel W . Then the interfacial stress tensor Π , being effectively a product of colour gradients, has order twice that of W . Because we are using the Wendland quintic spline, the order of Π will be 10.

Because the corrected SPH gradient can only exactly reproduce the constant gradients of linear functions, we cannot successfully use [equation \(5.38\)](#) to calculate the divergence of Π to get our force. The surface correction is in effect a kernel consistency correction up to first order. Liu & Liu [69] detailed a method for correcting particle inconsistencies up to first order, but which can be extended to arbitrary order n . Since near the kernel inconsistency near the boundary is always accompanied by a corresponding particle inconsistency one could apply their method as well to correct for the missing support. Their method in effect enforces the constraints of normalization, symmetry, and so forth up to the n -th order on the smoothing kernel W by solving a set of linear equations. This method has been shown to work well in $2d$ up to first order. The downside however is that the set of equations has size

$$\sum_{i=0}^n d^i = \frac{d^{n+1} - 1}{d - 1}. \quad (5.74)$$

For first order this is 3 equations in 2D and 4 equations in 3D (corresponding to the functions values and its gradients in each dimension). This method, further, tells us that we can only correct up to the order of the smoothing kernel (here 5). If we had a 10-th order smoothing kernel and wished to correct the missing support using this method we would have to solve a set of 2047 linear equations for each SPH particle. Clearly this method will be too cumbersome and computationally costly to allow us to capture the CSF forces correctly.

5.7.4 Curvature Based Surface Force

Instead of using the CSF another method for calculating surface tension force in SPH involves working directly with the curvature. As in [section 3.3](#) the interface normal can be calculated using the colour gradient:

$$\hat{\mathbf{n}} = \frac{\nabla C}{|\nabla C|}.$$

The normals are therefore only defined in a region $2h$ wide either side of the interface (because elsewhere $|\nabla C| = 0$). Still, where a colour gradient exists one can calculate the divergence of the normal to give the curvature, as in [section 3.3](#). Using the simple SPH gradient this is

$$\nabla \cdot \hat{\mathbf{n}}_i = \sum_j (\hat{\mathbf{n}}_i - \hat{\mathbf{n}}_j) \cdot \nabla W_{ij} V_j. \quad (5.75)$$

Taking the difference of the normals $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{n}}_j$ ensures that even in the presence of particle disorder two normals with the same direction contribute nothing to the force experienced by particle i . In a similar fashion to the particle correction which leads to [equation \(3.19\)](#) we can correct the divergence estimator for particle disorder. As detailed in Adami et al [1] and Breilinger et al [19] the ‘corrected divergence’ surface tension force is written;

$$\nabla \cdot \hat{\mathbf{n}}_i = d \frac{\sum_j (\hat{\mathbf{n}}_i - \hat{\mathbf{n}}_j) \cdot \nabla W_{ij} V_j}{\sum_j r_{ij} |\nabla W_{ij}| V_j}. \quad (5.76)$$

The behaviour of the interface can further improved by weighting the colour used to calculate ∇C_i^i according to the density of the fluid phases;

$$\tilde{c}_{ij}^i = \frac{\rho_j}{\rho_i + \rho_j} c_i^i + \frac{\rho_i}{\rho_i + \rho_j} c_j^i. \quad (5.77)$$

This modification has the effect of shifting the surface tension force primarily onto the heavier phase, which, because the SPH particles are less easily accelerated, improves the spurious velocities at the interface. The force given by [equation \(5.76\)](#) is applied only between fluid phases. No interaction with solid boundaries is specified by [1, 19] meaning that any droplet will naturally adopt a contact angle of $\theta = 90^\circ$. Extra work is required to make the droplets in contact with walls behave as they should.

5.7.5 Contact Points and Contact Angles

With the CSF model the corrected gradient misestimates the true divergence of the stress tensor when using the boundary integral method. This leads to the fluid adopting the wrong equilibrium shapes and having incorrect dynamics. Below are two potential methods for simulating correct contact angles

5.7.6 Method 1: Potential Energy Formulation

With curvature based surface tension models there is no force felt between the fluid phases and the walls (since there is no curvature) meaning that any droplets will tend to a contact angle of $\pi/2$ which corresponds to minimising the surface area of the fluid-fluid interface. To account for the surface energy of the fluid-solid interfaces we apply an attractive force to SPH particles belonging to the partially wetting phase l . Normally the l - s boundary represents an energy penalty, albeit a smaller penalty than the l - g interface. Since we are applying an attractive force the interface actually represents an energetically favourable state thus α_{ls} will appear with the opposite sign to usual in Young's equation; $\alpha_{gl} \cos(\theta) - \alpha_{sl} = \alpha_{gs}$.

In order to achieve the desired contact angle we must ensure that the energy contained at the l - s boundary per area corresponds to the correct surface energy α_{ls} . If a particle at distance r from the walls feels an acceleration $a(r)$ into the wall (that is, $\mathbf{a} \cdot \hat{\mathbf{n}} < 0$) then the potential of that particle is

$$V(r) = \int_{\infty}^r a(x) dx. \quad (5.78)$$

The acceleration is chosen such that it acts over a small distance of $2h$ only, meaning that for particles with $r \geq 2h$ the potential is zero and for $r < 2h$ it is negative. The total energy in the system due to this attractive wall force is given by the volume integral $U_{tot} = \int_{\Omega} V(\mathbf{r}) d\mathbf{r}$. We know that the surface tension α_{ls} is effectively the energy per unit area of the surface meaning that

$$\alpha_{ls} = \int_0^{\infty} \rho(r) V(r) dr. \quad (5.79)$$

which since our fluid is (approximately) incompressible is

$$\alpha_{ls} = \rho_0 \int_0^{\infty} V(r) dr. \quad (5.80)$$

To move forward with this we need a concrete form for the acceleration $a(r)$. Let $a(r)$ be a linear function over the interval $[0, 2h]$ and zero otherwise;

$$a(r) = \begin{cases} C \left(1 - \frac{r}{2h}\right) & \text{if } 0 \leq r \leq 2h \\ 0 & \text{otherwise.} \end{cases} \quad (5.81)$$

Substituting this form for $a(r)$ into [equation \(5.78\)](#) gives us

$$V(r) = \begin{cases} C \left(r \left(1 - \frac{r}{4h}\right) - h\right) & \text{if } 0 \leq r \leq 2h, \\ 0 & \text{otherwise} \end{cases} \quad (5.82)$$

which with [equation \(5.80\)](#) implies that for the surface energy to match α_{ls} we require

$$C = -\frac{\alpha_{ls}}{\rho} \frac{2}{3} h^2. \quad (5.83)$$

This integral assumes that there is a continuum in the range $0 \leq r \leq 2h$ when in fact there are actually only $N = \lfloor 2s \rfloor$ SPH particle layers within $2h$ of the wall. This means that if we

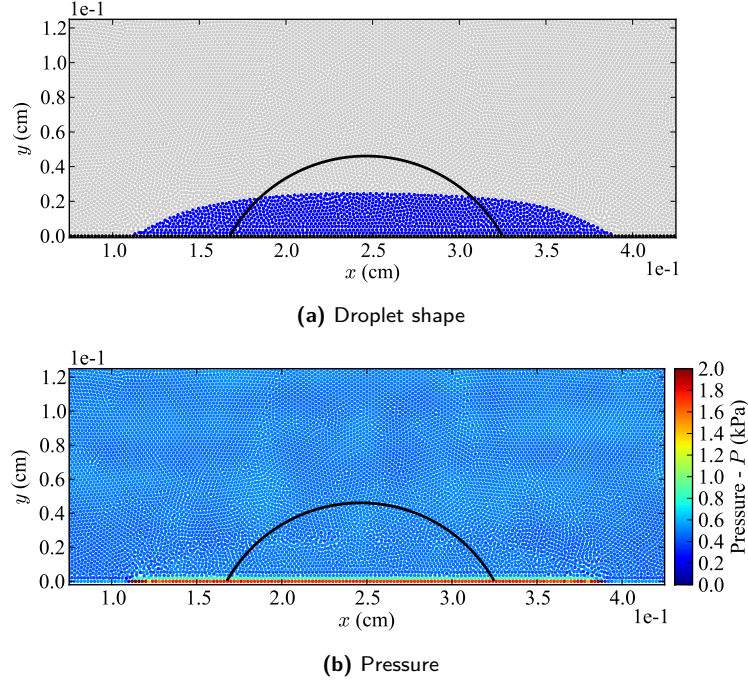


Figure 5.27: Applying the potential energy method as detailed leads to the droplet overshooting its specified contact angle. Additionally the pressure near the wall is far higher than away from the wall due to the SPH particle being pulled closer to the boundary.

apply the acceleration $a(r)$ to these particles then the actual potential energy felt by the SPH particles will be significantly different from [equation \(5.80\)](#).

To counter this we can instead use the sum

$$\alpha = \rho_0 \sum_{n=1}^N \Delta x V(n\Delta x) \quad (5.84)$$

to calculate the potential caused by the force into the wall. This gives us a new value for C :

$$C = -\frac{\alpha_{ls}}{\rho_0 h^2} \frac{s}{\eta(s)} \quad (5.85)$$

where

$$\eta(s) = -Ns + \frac{1}{2}N(N+1) - \frac{1}{4s} \frac{1}{6}N(N+1)(2N+1),$$

and as expected $\lim_{s \rightarrow \infty} s/\eta(s) = \frac{3}{2}$ to match the continuum version.

Now, by applying the force with magnitude given by [equation \(5.85\)](#) we should be providing the system with an energy advantage of the correct size for the surface energy (per unit area) of α_{ls} . Therefore it was expected that this force would pull the fluid droplets to the correct equilibrium contact angle. Unfortunately this is not the case. The additional force felt by the particles near the wall leads to the particles rearranging themselves so that more of them can be closer to the wall while still balancing the resulting pressure force as is shown in [figure 5.27b](#). This disruption of the particle position seems to allow the SPH particles of the partially wetting

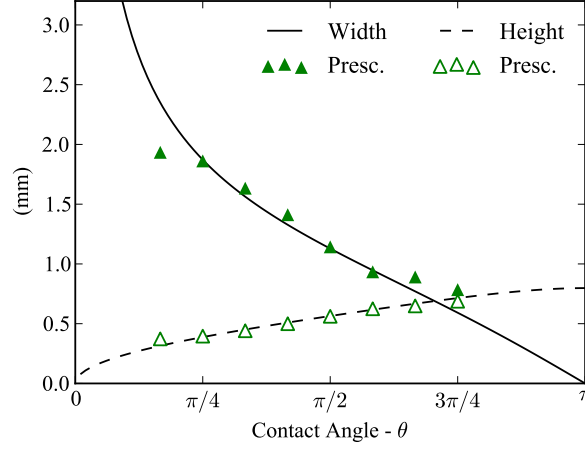


Figure 5.28: Prescribing the normals for the two phases directly, which are used by the surface force calculations, gives generally the correct contact angle but does suffer from some issues. At low contact angles it fails to reproduce the correct profile, this is mainly due to finite resolution effects, and for $\theta \gg 90$ the particle positioning at the interface differs from the expected behaviour.

fluid to slide closer to the wall and under the non wetting phase. This leads to the droplet adopting a far smaller contact angle than it should based upon the value of α_{ls} . To counter this effect a different form for $a(r)$ was investigated which had a repulsive element at short range, but this form also suffered from the same issues. The amount of the force can be adjusted ‘by hand’ so that the droplet adopts the correct contact angle but the factor by which one must adjust the force is highly depended on the specified contact angle and the speed of sound being used and so it seems that this is not an especially feasible method for handling contact forces with BIM walls.

5.7.7 Method 2: Prescribed Normals

Since the curvature based approach works directly with the normals we can follow the method of [19] and set the normals to what they should be if the fluid were at its equilibrium contact angle θ , i.e.

$$\hat{\mathbf{n}}_{\text{eq}} = \hat{\mathbf{n}}_{\text{wall}} \sin(\phi) + \hat{\mathbf{t}}_{\text{wall}} \cos(\phi). \quad (5.86)$$

where $\phi = \frac{\pi}{2} - \theta$. The unit vectors $\hat{\mathbf{n}}_{\text{wall}}$ and $\hat{\mathbf{t}}_{\text{wall}}$ are the wall normal and tangent respectively. The inward wall normal is known directly from the shape of the wall and is completely specified, the tangent here is calculated by projecting the gradient of colour onto the wall. So for a particle i we have

$$\hat{\mathbf{t}}_{\text{wall},i} = \frac{\nabla C - (\nabla C \cdot \hat{\mathbf{n}}_{\text{wall}})\hat{\mathbf{n}}_{\text{wall}}}{|\nabla C - (\nabla C \cdot \hat{\mathbf{n}}_{\text{wall}})\hat{\mathbf{n}}_{\text{wall}}|}. \quad (5.87)$$

To avoid a sharp jump in the normals as seen by the fluid, the prescribed normal is weighted so that at distances $\Delta x \leq x < 2h$ the normal which is prescribed is a mixture of the actual

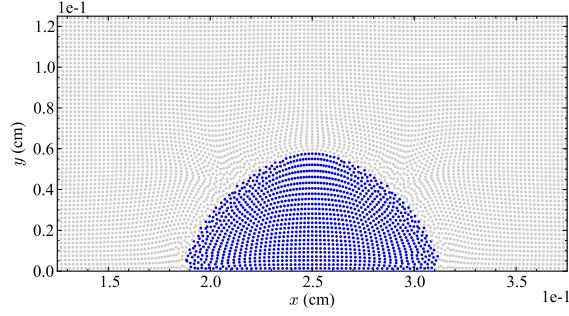


Figure 5.29: Prescribing the normals used for the surface tension calculations leads to a step in the particle arrangement while the droplet is spreading due to the mismatch between the actual normal and the prescribed normal.

normal and the equilibrium normal;

$$\hat{\mathbf{n}}_{\text{prescribed}} = f \hat{\mathbf{n}}_{\text{eq}} + (1 - f) \frac{\nabla C}{|\nabla C|}, \quad (5.88)$$

where

$$f(x) = \begin{cases} 1 & x < \Delta x \\ \frac{x - \Delta x}{2h - \Delta x} & \Delta x \leq x < 2h \\ 0 & x \geq 2h \end{cases} \quad (5.89)$$

The advantage of prescribing the normals in this way is that since we only calculate the interface force based on the two fluids, the wall does not enter into the calculation and we are thus able to correctly calculate the resulting force.

Exactly at the contact point the normals used do not match the normal arising from the actual SPH particle arrangement, which leads to a slight ‘step’ in the particle arrangement as seen in [figure 5.29](#). This step is present predominately as the droplet begins to spread from its rectangular initial state. As the simulation progresses the droplet settles into the correct shape with no step observed for $\theta < 90^\circ$.

We can see from [figure 5.28](#) that for a range of equilibrium contact angles the droplet assumes the correct shape but at low θ and $\theta > 90^\circ$ there are discrepancies. For low θ ($\lesssim 30^\circ$) the droplet does not reach the correct angle due to issues of finite resolution. When the droplet is very spread out, due to the particle based nature of SPH, the very edge of the droplet height consists of a single particle meaning we cannot spread any further than this.

The discrepancy for $\theta > 90^\circ$ is surprising since this amounts to specifying a contact angle $\theta' < 90^\circ$ for the bulk fluid phase, and we have seen that for $30^\circ < \theta < 90^\circ$ the droplet assumes the correct shape. The only difference in this cases is that by giving the droplet a

large contact angle we are actually increasing the pressure inside the droplet due to the Laplace pressure

$$\Delta P = \frac{\alpha_{lg}}{r(\theta)}, \quad (5.90)$$

where r is the droplet radius as given by [equation \(5.68\)](#). Therefore for a large contact angle the surface force is having to overcome an increasing pressure and, it seems, it is insufficient to succeed.

Despite the inaccuracies this method successfully achieves the correct contact angle for a useful range of θ and is therefore a useful technique for combining surface forces with the boundary integral method.

5.8 Conclusions

In this chapter we have highlighted issues involved with implementing reflected boundary conditions for curved objects in SPH. It was shown that when using geometric objects care must be taken to correctly renormalize the weightings for both the fluid particle and the corresponding reflected particle and that without accounting for the ‘volume-factor’ the simulations are demonstrably less accurate. Including objects as geometrical representations is desirable as often one would like a simplified setup to mimic experiments or to remove extraneous factors from consideration. Further, this framework could easily be extended to include more complicated objects such as superquadrics.

NURBS curves and surfaces are the industry standard for much CAD and technical drawing work. Therefore the fact that they can be directly included in a practical manner into an SPH implementation is a very desirable feature. The first half of this chapter examined the issues involved with their application as boundaries in SPH using reflected particle boundaries. It was shown that in 2D that the Newton-Raphson method provides a very quick way of searching the parameter space and locating closest points, although care must be taken to detect the occasional divergences which occur as part of the method. In 3D it is expected that the same method should also work well.

Next the boundary integral method (BIM) was investigated, this is based on a recently published method from Kulasegaram et al and Ferrand et al [\[37, 63\]](#). The form used in this thesis is derived in a novel way from the zeroth order complete kernel [equation \(3.16\)](#) and implementation using boundary particles was discussed. While the method performs well for pressure and viscosity it falls down when handling surface tension forces and contact angles due to the higher order nature of these forces. While in principle these forces are entirely specified by what is happening inside the simulation domain correcting for the missing support in this case presents a unique challenge. We investigated two potential methods for combining contact angles successfully with the BIM boundaries. The first, potential energy formulation, was found to disrupt the usual particle arrangement, which is caused by pressure between the SPH particles. This lead to the droplets simulated assuming not only the wrong contact angle but also having extended leading edges. The second method, prescribed normals, was found to be successful for a useful

range of contact angles but this method too has limitations not seen with the CSF model combined with reflected ghost particle boundaries.

Chapter 6

Numerical Stability & Accuracy

The exact mass conservation and automatic handling of topological changes and interface tracking make SPH a suitable method for simulating flows in heaps. Mostly these are properties one can also find with the Lattice Boltzmann method, which is another popular method for simulating (multiphase) flows in porous media. One advantage of SPH over the Lattice Boltzmann method is that the physical parameters of the fluid(s) being simulated are direct parameters of the equations of motion and one is not bound by the resolution of the simulation grid.

Beyond the parameters that are set by the fluids' physical properties, the SPH method is influenced by the numerical speed of sound, c_0 , chosen. Since the SPH method used here is not truly incompressible, but only 'weakly compressible', sound waves can propagate through the fluid. The value of c_0 has several effects on the behaviour of the simulated fluids that are relevant to this thesis.

The interfaces of fluids, for which surface tension is included through the CSF model (see [section 3.3](#)), are subject to unwanted velocities, even when the droplets are sessile. This effect is known as spurious velocity currents. We have found that the numerical speed of sound, and background pressures, have a large effect on these spurious velocities. In addition the area over which the surface force is applied has an effect on the spurious velocities.

Secondly, the speed of sound is directly related to the compressibility of the fluid in such a way that increasing c_0 decreases compressibility. This effect is well known and an investigation by Becker & Teschner [8] suggested a rule of thumb for setting c_0 ; it should be set to at least 10 times the maximum fluid velocity in the system. Despite the fact that higher c_0 gives a lower compressibility *and* seemingly better interfacial behaviour one should not increase it too far. In this chapter we investigate the effect of the numerical speed of sound on the SPH solutions as well as briefly discussing the spurious velocities.

6.1 Spurious Velocities

One problem which is well known in the literature when simulating interfacial tension with the CSF model is that of spurious velocity currents¹. At fluid interfaces, and in particular triple points & contact lines, systems which should be in equilibrium show unphysical non zero velocities near the interfaces. Lafaurie et al showed that the largest size of these velocities is given by

$$|\mathbf{u}_{s,\max}| \propto \frac{\alpha}{\mu},$$

where the constant of proportionality is of order 10^{-2} [64]. Additionally Brackbill & Kothe [17] showed that the velocities can be improved by reducing the width of the interface region. An

¹Sometimes called 'parasitic velocities'.

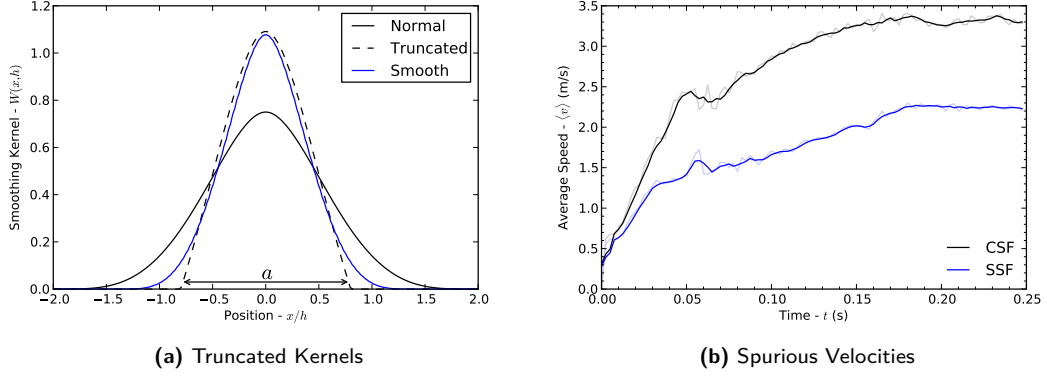


Figure 6.1: (left) The truncated and narrowed kernels as used for the ‘sharpened surface force’ calculations. (right) The spurious velocities evolving in a static droplet which is initially approximately circular.

in depth discussion of this issue as pertaining to the Volume of Fluid (VOF) scheme can be found in A. Raeini’s PhD thesis [87].

The fact that the spurious velocities can be reduced by decreasing the interface thickness has relevance for SPH since typically the interface region is taken to be $4h$ meaning there are $\approx 4h/\Delta x = 4s$ particles across the interface region. Therefore as s increases the spurious velocities will be more of an issue. To counter this, we investigated the use of a ‘sharpened surface force’ (SSF) where the interface force is redistributed over a region $2a < 4h$ thick. To achieve this the values for $|\nabla C_i|$ in the calculation of the interfacial stress were modified so that they fall to zero before $2h$. Two subtly different methods were used to modify $|\nabla C_i|$.

The first method involves simply truncating the smoothing kernel $W(r, h)$ and then renormalizing so that the integral is still unity:

$$K_1(r, a) = \begin{cases} K_a W(r, h) & \text{if } r < a \\ 0 & \text{otherwise} \end{cases}, \quad (6.1)$$

where

$$K_a = \int_{-a}^a W(x, h) dx.$$

The second method involves inverting $\int W$ to calculate the effective distance of particle i from the interface, i.e.

$$r_i = W_h^{-1}(|\nabla C_i|)$$

then recalculating the weighting $|\nabla C_i|$ using this distance and a narrower kernel; $|\nabla C_i|' = W(r_i, a)$.

Figure 6.1b shows the mean velocity for a droplet which is initially at rest and approximately circular². It is clear that for the SSF simulation the spurious velocity currents are smaller than in the usual case. It seems, however, that this is an artefact of spuriously lower surface tension

²The particles are initially arranged on a grid so some rearrangement of particles is expected.

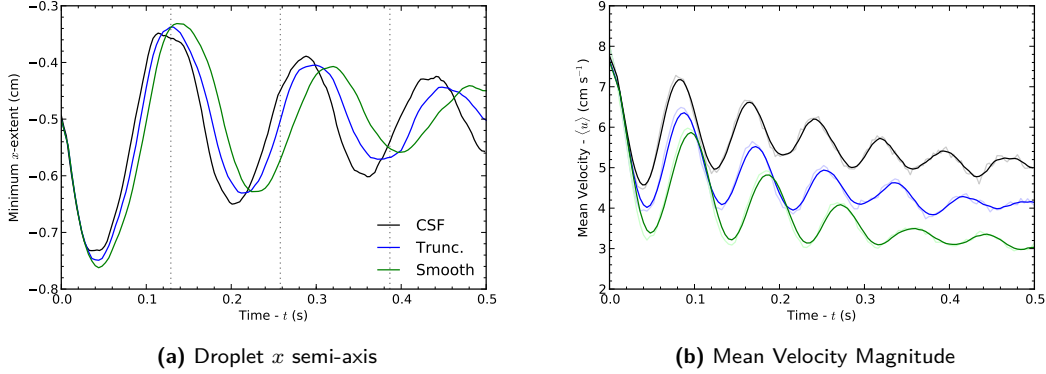


Figure 6.2: The droplet's minimum extent in the x direction (x Semi-axis) and the mean velocity within the fluid droplet show the difference between the usual CSF and the two SSF simulations.

in the SSF simulations, which is highlighted by simulating a dynamic system.

As in [49] we consider an oscillating water droplet with initial divergence free velocity given by $\mathbf{u}(0) = 12.5 (x, -y)^T$. The oscillation period of a droplet in free space is

$$T = 2\pi \sqrt{\frac{\rho R^3}{6\alpha}}, \quad (6.2)$$

where R is the droplet radius, which in this case is 0.5 cm giving $T = 0.128$ s. Our droplet however is oscillating in air so this will only be approximately correct.

Shown in figure 6.2 are the minimum extent of the droplet in the x direction and the mean velocities within the droplet plotted against time, for each of the three simulations. It is clear that the behaviour, while similar, is subtly different. For the base case (CSF) the oscillations match closest the period as given by equation (6.2) (indicated on the graph as grey dotted lines). The two SSF techniques are close to this initially but after only a few cycles it is apparent that they are under estimating the frequency of oscillation with the smoothed kernel performing worse than the truncated kernel. Because ρ and R were the same for all three simulations, a lower frequency indicates a lower effective surface tension.

This discrepancy between specified and measured surface tension is most likely due to the lower number of particles within the interface region of width $2a$ (as compared to $2h$). Having fewer particles causes larger errors in our numerical approximations. This suggests that perhaps taking a higher s might lead to better performance with the SSF interfaces.

6.2 Pressure Induced Locking

The numerical speed of sound is linked to the compressibility of the fluid. We find however that increasing it too far can lead to the SPH particles ‘locking up’ and forming rigid structures (at overpressure) which cause the SPH solution to deviate from the true analytical solution. In the

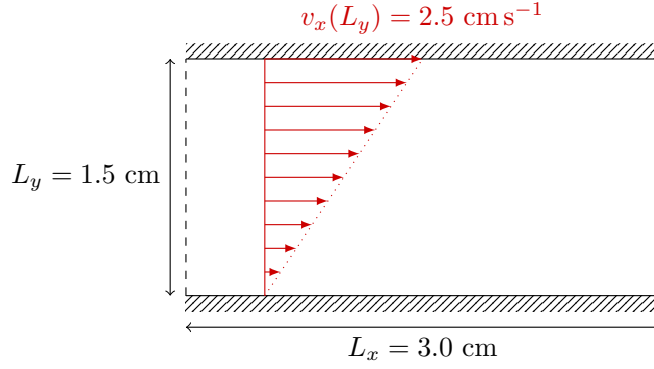


Figure 6.3: Simulation setup for flow between two plates. Periodic boundary conditions were applied in the x direction, the top plate moves in the $+x$ direction with velocity 2.5 cm s^{-1} .

rest of the chapter we investigate this effect.

The compressibility of a fluid is the relative change in volume for a given change in pressure. Mathematically it is written

$$\beta = -\frac{1}{V} \frac{\partial V}{\partial P}, \quad (6.3)$$

which we can rearrange to get in terms of the density ρ and the equation of state $P = u(\rho)$;

$$\beta = \frac{1}{\rho} \left(\frac{du}{d\rho} \right)^{-1}. \quad (6.4)$$

The speed of sound for a compressible fluid is

$$c_0 = \sqrt{\frac{du}{d\rho}}, \quad (6.5)$$

which can be substituted into [equation \(6.4\)](#) giving

$$\beta \propto \frac{1}{c_0^2} \Rightarrow c_0 \propto \frac{1}{\sqrt{\beta}}.$$

The CFL timestep condition relates c_0 to Δt so we can see that $\Delta t \propto \sqrt{\beta}$. With modern computing power and efficient parallel code it can be tempting to simply increase c_0 to some large value to attempt to get a very low compressibility and to improve interface sharpness. This can, however, itself cause issues which in extreme cases cause large deviations of the SPH output from the true solution.

6.2.1 Shear Flow Example

To demonstrate the issues with increasing c_0 too far we consider a simple test case for which the exact analytical solution is known; flow between two infinite parallel plates. The 2D simulation setup is shown in [figure 6.3](#). The fluid is confined between two plates width $L_y = 1.5 \text{ cm}$ apart. The upper plate moves with a velocity $\mathbf{u} = (u_0, 0)^T$ with $u_0 = 2.5 \text{ cm s}^{-1}$. Periodic boundary

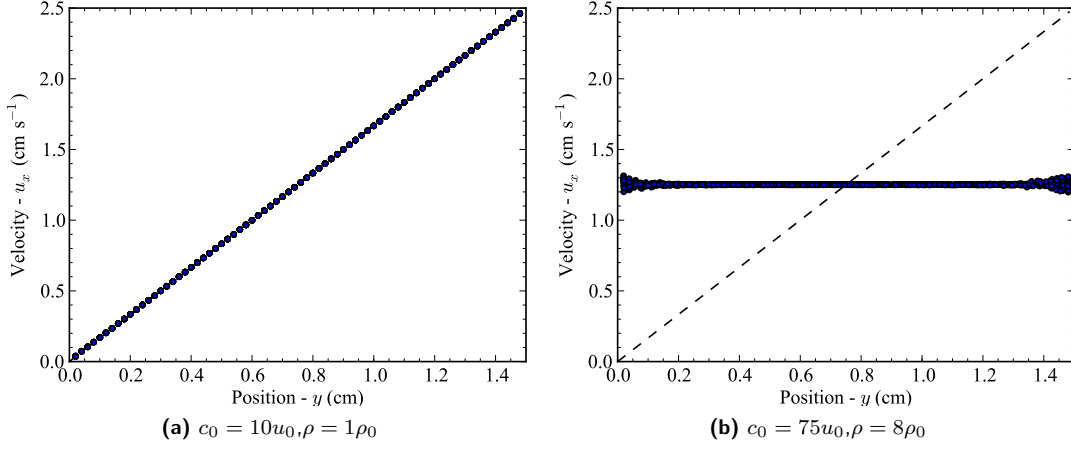


Figure 6.4: Solutions at $t = 10.0$ s for two different parameter settings. Plotted is the x velocity component u_x against height y ; the SPH solution is shown as points with the analytical solution as dashed. Clearly the simulation with the highest bulk modulus suffers extreme failure to capture the correct profile due to pressure locking.

conditions are applied in the x direction with period length $L_x = 3$ cm. The analytical solution for this flow is a linearly increasing x -velocity with height y ;

$$u_x(y) = \frac{u_0}{L_y} y \quad (6.6)$$

and $u_y = 0$ everywhere.

For the simulations in this section we use a ‘base’ fluid with density $\rho_0 = 1000 \text{ kg m}^{-3}$, viscosity $\mu_0 = 0.01 \text{ kg s}^{-1} \text{ m}^{-1}$ and speed of sound $10u_0 = 25 \text{ cm s}^{-1}$. The base speed of sound is chosen as such because the condition recommended in [8] is that c_0 be $\sim 10u_{\text{max}}$. The density, viscosity and speed of sound were each varied from their base values, as well as the spatial discretization length h , in order to study the effect of pressure locking. The output from two of these simulations is shown in figure 6.4 where the velocity u_x is plotted against y after running for $t = 10$ s. The analytical solution, equation (6.6), is shown as a dashed line.

The two plots shown in figure 6.4 differ only in the bulk modulus appearing in the equation of state, $B = \rho_0 c_0^2$. The second plot has B a factor of 450 higher. Figure 6.4b shows that such a high bulk modulus destroys SPH’s ability to correctly recreate the flow properties. Rather than the velocity varying linearly throughout the range of y , the system is instead moving with an approximately constant velocity throughout the domain with two sharp jumps in velocity at either end. This indicates that the system is moving as a whole, like a rigid body, rather than as a fluid. The high speed of sound leads to a faster varying pressure with density through equation (3.28); these high pressure changes cause the system to lock into a hexagonal close packing. The SPH particles become so tightly locked into this configuration that the laminae can no longer flow past one another as they should.

For simulations in the transition region between accurate and fully locked we see crystalline regions of SPH particles which persist over short time scales but are small compared to the

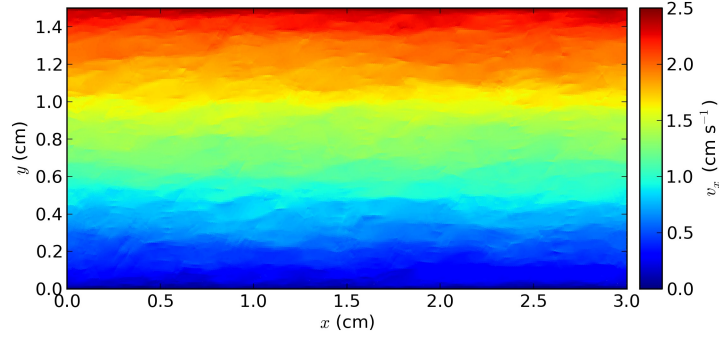


Figure 6.5: The velocity profile for a simulation with $c = 50c_0$ and $h = 0.5h_0$ and all other parameters having their base values. Some locking is seen but it is not complete; the velocity profile approximately matches the analytical but some artefacts are clear.

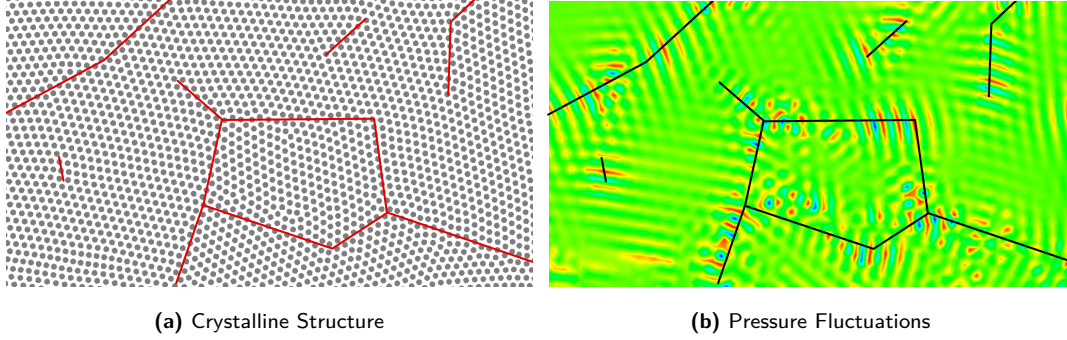


Figure 6.6: The particle positions (zoomed view) for simulation with $c = 50c_0$ and $h = 0.5h_0$, the lines are drawn in order to highlight the boundaries between crystalline regions.

system. These regions seem to be able to move past each other so the simulation becomes inaccurate, but does not break down completely, see [figure 6.5](#). The edges of the regions are characterized by ‘dislocations’, which manifest themselves as large localized fluctuations in the pressure field. An example of the crystalline structure and the corresponding pressure fluctuations are shown in [figure 6.6](#).

This locking behaviour can be confirmed by examining the spatial correlation of SPH particle positions. $G(r)$ is the expected number of particles at a distance r from each other particle.

$$G(r) = \frac{1}{2N} \sum_{i=1}^N \sum_{j=1}^N \delta(|\mathbf{x}_i - \mathbf{x}_j| - r) \quad (6.7)$$

We can see from [figure 6.7](#) that at low speeds of sound the neighbours are found mostly at multiples of Δx , with an approximately constant background at distances greater than Δx . This is easily explained; the other particles in a particle’s lamina sit at multiples of Δx away. The particles in the neighbouring laminae are at approximately uniformly distributed because the laminae slide past each other but, because the laminae are spaced Δx from one another in the y direction, the closest approach of a particle from a neighbouring lamina is Δx . At

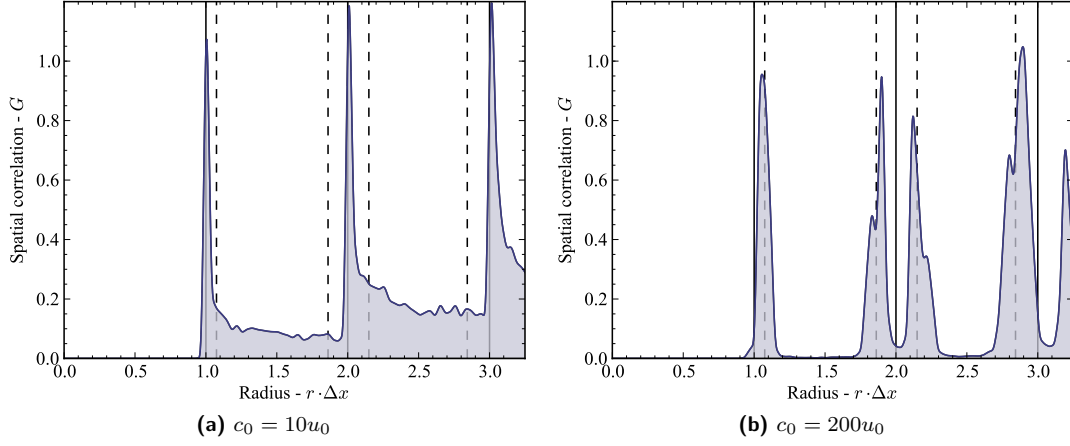


Figure 6.7: The expected number of neighbours G at distance r for two different speeds of sound. The solid black lines correspond to spacings at multiples Δx and the dashed lines correspond to the spacings in a regular hexagonal packing. It is clear that at the higher c_0 the SPH particles have locked in a hexagonal structure as indicated by $G(r)$.

higher speeds of sound we see neither the constant background nor the peaks at multiples of Δx . Instead there are large peaks coinciding with where we would expect to see particles if they were on a completely regular hexagonal close packing (indicated in [figure 6.7b](#) by the dashed lines).

[Figures 6.5 to 6.7](#) clearly illustrate what is happening for the systems with high bulk modulus, but understanding when this issue is likely to arise is also important. Possible quantities which could affect this jamming are density - ρ , viscosity - μ , speed of sound - c_0 , smoothing length h and the time step - Δt . The smoothing length and the timestep depend on one another through the acoustic CFL condition and are, thus, not independent quantities so we will consider only the timestep Δt . To quantify the deviation from the analytical solution we again use the RMS error;

$$E_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^N (u_{x,i} - u_x(y_i))^2} \quad (6.8)$$

with the sum being over all SPH fluid particles. The units of E_{RMS} are those of velocity. There are five quantities of interest here (ρ , μ , c_0 , Δt and E_{RMS}) and three dimensions (time, mass and length) giving two possible dimensionless groups. The dimensions (see [table 6.1](#)) combine to give

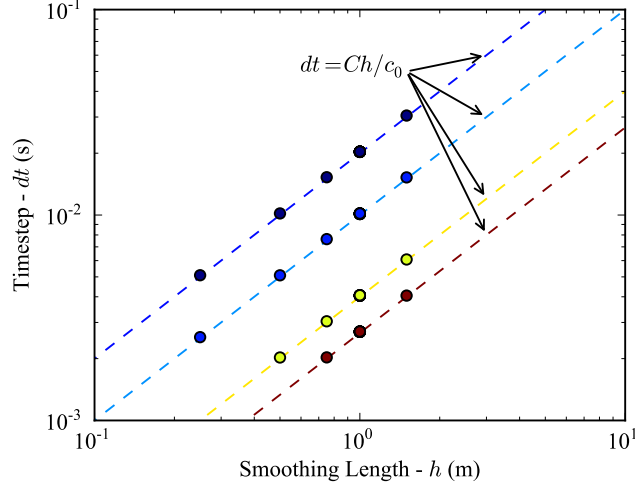
$$\pi = \frac{\rho c_0^3 \Delta t}{\mu E_{\text{RMS}}} \quad (6.9)$$

as one possible dimensionless group (up to a power). The other possible group does not contain the quantity of interest E_{RMS} . From [equation \(6.9\)](#) we get directly

$$E_{\text{RMS}} \propto \frac{\rho c_0^3 \Delta t}{\mu}. \quad (6.10)$$

Table 6.1: Dimensions of relevant quantities.

Quantity	Dimensions
E_{RMS}	LT^{-1}
ρ	ML^{-3}
c_0	LT^{-1}
Δt	T
μ	$ML^{-1}T^{-1}$

**Figure 6.8:** The simulation timestep plotted against smoothing length h for several speeds of sound c_0 . The dashed lines represent the value of the acoustic CFL constraint for each c_0 .

Because of the adaptive time-stepping used, Δt is not set as a parameter of our simulations, rather it is calculated to obey the necessary stability constraints (section 3.4.1). The acoustic CFL condition states that to remain stable the simulation must have

$$\Delta t < \min_i \left\{ \frac{h}{c_0 \sqrt{(\rho_i^*)^{\gamma-1}}} \right\}. \quad (6.11)$$

In this case we can confirm that this is the tightest constraint on the timestep by plotting Δt against h for several speeds of sound and confirming they obey $\Delta t \approx Ch/c_0$ where $C = 0.2$ is the convergence factor used in the time-step calculations. Figure 6.8 shows that this is indeed the most stringent constraint and that we can set $\Delta t \propto h/c_0$. Substituting this into equation (6.10) we get

$$E_{\text{RMS}} = k \frac{\rho c_0^2 h}{\mu}, \quad (6.12)$$

for some constant k . We can equivalently consider h or Δt to be the quantity of interest.

Examining the RMS error against $\rho_0 c_0^2 h / \mu$, as shown in figure 6.9, we can see that equation (6.12) is, at least to a first approximation, a reasonable estimate of how the error behaves. The sloped dotted line represents a best fit of the equation (6.12) with the constant of proportionality k as the free parameter. The horizontal dotted line shows the RMS error which would be generated by a purely plug flow; that is $v(y) = v_0/2$. There is no theoretical maximum value

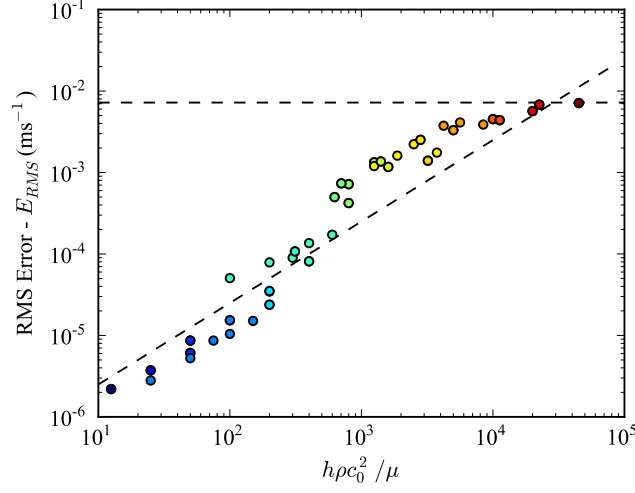


Figure 6.9: The RMS error between the SPH solution and analytical solution plotted against $h\rho c_0^2/\mu$. The dashed line corresponds to a linear relationship. The combination of parameters $h\rho c_0^2/\mu$ gives the closest to a linear relationship with the error.

for E_{RMS} but the plug flow situation represents the grossest failure a functioning simulator should be capable of achieving.

6.2.2 Balance of Forces

The relationship between the simulation parameters and the RMS error, [equation \(6.12\)](#), while making dimensional sense, was not derived from any physical considerations other than basic dimensionality. It is, however, relatively easy and intuitive to show that the error arises when the balance between the pressure forces and the viscous forces becomes too skewed towards the pressure. Considering the forces additionally allows us to derive the dependence of E_{RMS} on the smoothing factor, s , which is dimensionless and thus not amenable to dimensional analysis. It is possible to show that the pressure forces felt by the SPH particles can be approximated to first order as being given by a quadratic potential;

$$\phi(\varepsilon) \propto \frac{\rho_0 c_0^2}{h^2 s^{2d}} \varepsilon^2, \quad (6.13)$$

where ε is the particle's displacement from its equilibrium position in a regular lattice of SPH particles. The derivation (and full form) of ϕ is detailed in [appendix A.3](#). The force density felt by an SPH particle due to pressure is given by $-\nabla\phi$. Meaning we have

$$f_{\text{press.}} \propto \frac{\rho_0 c_0^2}{h^2 s^{2d}} \varepsilon.$$

Next, the force density due to viscosity is

$$f_{\text{visc.}} = \mu \nabla^2 \mathbf{u},$$

a finite difference approximation for which could be written

$$f_{\text{visc.}} = \mu \frac{\Delta u}{h^2},$$

for some Δu which has units of velocity. The ratio of these two forces then is given by

$$\pi = \frac{f_{\text{press.}}}{f_{\text{visc.}}} \propto \frac{\rho_0 c_0^2 \epsilon}{s^{2d} \mu \Delta u}, \quad (6.14)$$

after cancellation. The distance a particle must move out of its quadratic potential for the lamina to slide past each other is clearly some fraction of the spatial discretization length, i.e. $\epsilon \propto h$, implying finally that

$$\Delta u \propto \frac{\rho_0 c_0^2 h}{\mu s^{2d}}. \quad (6.15)$$

Thus confirming the intuitive reason for the breakdown of the SPH solutions as being the pressure force stopping the lamina sliding past each other. Clearly [equation \(6.15\)](#) is not a complete description because it implies the error can be increased without bound by increasing ρ , or even more quickly by increasing c_0 . However, the highest error practically achievable by a reasonable simulation is that of plug flow for which there is a finite non-zero value of E_{RMS} .

6.2.3 Background Pressures

Many authors introduce a background pressure into their simulations with the aim of combating tensile instability³ but its value must be chosen carefully; too low and it provides no benefit, too high and we see the same pressure locking. The background pressure is introduced simply through adding a constant to the equation of state;

$$u(\rho) = \frac{\rho_0 c_0^2}{\gamma} \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right) + P_0 \quad (6.16)$$

where P_0 is the background (or atmospheric) pressure. Note that for free surface flows P_0 must be zero so this can only be an issue for closed-system type simulations which can be at overpressure.

Shown in [figure 6.10](#) are the results from an interfacial SPH simulation at two different background pressures. One can clearly see that the background pressure is affecting the simulations' outputs. At the high background pressure the the interface suffers from much reduced spurious velocities which can be seen in the lack of noise in the relative density field at the interface. However, the particle locking caused by the higher P_0 has prevented the interface from taking on the correct contact angle of $\theta = 90^\circ$. At the lower P_0 the droplet has assumed the correct shape but there are quite large spurious velocities.

³See [section 3.2.1](#)

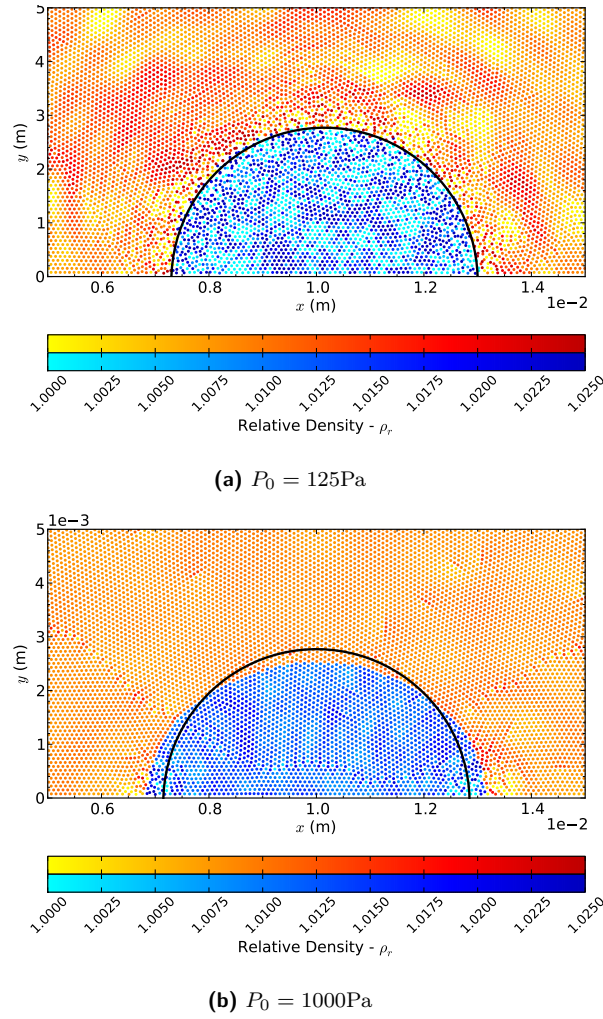


Figure 6.10: Output of interfacial flow simulation at $t = 5$ s at two background pressures P_0 showing that background pressure can strongly affect results. At the higher pressure the spurious velocities have been reduced but the interface has failed to take on the analytical shape indicated in black.

6.3 Conclusions

In this chapter we have investigated two aspects of the SPH method which are ancillary to the physical results, but are affected strongly by the parameters chosen for the simulations. While the results are fundamental to simulating closed systems with SPH, and in some sense quite basic, it is not something which has been reported in the literature.

In this thesis the primary motivation for increasing the speed of sound and / or the background pressure is to suppress the spurious velocities seen at fluid interfaces, but this can have disastrous effects on the simulation results. The rule of thumb that c_0 should be 10 times the maximum velocity in the system ensures an acceptably incompressible fluid without locking up the interfaces. But with no background pressure we find that the interfaces, while assuming the correct shapes readily, suffer from large spurious velocities. For the simulation scales presented

in the next chapter a background pressure of 250 Pa was found to be sufficient to improve the interface without being high enough cause any of the locking seen. The exact size of the background pressure necessary, however, does depend on the scale of the system being simulated and the specified speed of sound.

Chapter 7

Voronoi Surface Tension

The surface tension models detailed so far work nicely with SPH and fit naturally with the meshless nature of the numerical method well. However, they have some notable shortcomings. Firstly the issue of spurious velocities is a big limiting factor on the accuracy for smaller scale simulations and simulations where there are triple points, especially those not involving a wall. Secondly, while they can handle contact angles they cannot handle contact angle hysteresis. Lastly, they require the simulating of the fluids on either side of the interface which for our unsaturated simulations increases the computational cost quite significantly. In this chapter we will detail preliminary work towards a new surface tension method which should be able to remedy all of these issues. The issues arising with the normal smoothed surface tension models all stem from the lack of topology between the SPH particles. Because of this there is no well defined interface and we must work with colour gradients. If one had a well defined interface one could in principle ameliorate these problems. Therefore we develop a Voronoi based approach to define the topology between interfacial particles thereby providing access to the interface topology.

7.1 Triangulation Algorithm

To begin, a discussion of some properties of the Voronoi tessellation and its dual, the Delaunay triangulation is useful as they will be referred to later. The Voronoi tessellation for a set of N points divides the space up into the regions which are closest (under a given metric) to each point i . Thus associated with each point i is a region $\Omega_i \subset \mathbb{R}^d$ for which

$$\|\mathbf{x} - \mathbf{x}_i\|^2 < \|\mathbf{x} - \mathbf{x}_j\|^2 \quad \forall \mathbf{x} \in \Omega_i, j \in \{1, \dots, N\}, j \neq i,$$

under the usual Euclidean metric. The Voronoi tessellation is the dual of the Delaunay triangulation. For each triangle in the Delaunay triangulation, the circumcircle contains no other points. This is the defining feature of the Delaunay triangulation. The centres of each circumcircle coincide with the nodes of the Voronoi tessellation. Therefore obtaining the Delaunay triangulation allows us to compute the Voronoi tessellation.

Several algorithms exist for finding the Delaunay triangulation of a set of points, the best of which have $O(N \log N)$ runtime. If were to use such an algorithm to generate the triangulation (and thence the interface Voronoi tessellation) we would be compromising the runtime complexity of our SPH code, which is $O(N)$. However, we are only interested in the triangulation of the particles which form the interface. Knowing this fact, it is possible to develop an algorithm which can triangulate the interface in only $O(N)$ time, thereby maintaining the favourable scaling of the SPH method.

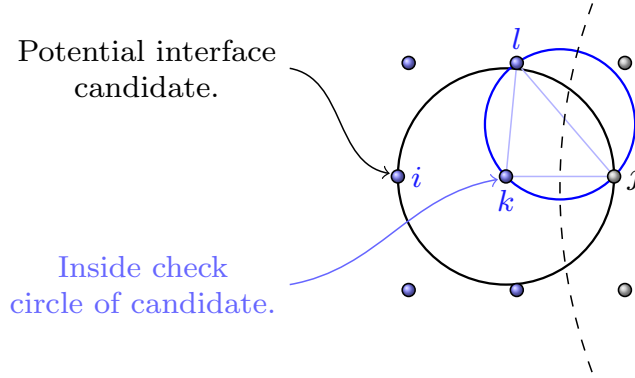


Figure 7.1: The particle i is identified as a potential candidate for being an interface particle, but we can see that the check circle which passes through i and j contains another particle k and so i cannot be an interface particle.

The algorithm proceeds by first finding a particle i which is close to the interface; identified by the fact it has a non-zero colour gradient. Next the closest particle j from another fluid phase is located. This pair (i, j) does not necessarily form part of the triangulation however. To check whether i is in fact part of the triangulation one can check that there are no particles inside the circle with radius $\|\mathbf{x}_i - \mathbf{x}_j\|/2$ and centre $(\mathbf{x}_i + \mathbf{x}_j)/2$, i.e, the smallest circle that passes through i and j . If a particle k is found inside this circle then i is a better potential candidate for being an interface particle and we now take k as the candidate. The new pair must still be from different phases. This checking continues until no particle is found in the check circle. Depending on the value of s used, this check typically requires only a single iteration to find the interface candidate.

Once we have identified the initial edge (k, j) the algorithm then checks all nearby particles (other than k and j) to find the particle l , for which the circumcircle (j, k, l) contains no other particles. This checking of nearby particles can be done quickly by using the linked cell grid (see [section 4.1.2](#)). This triangle is therefore part of the interface Delaunay triangulation. Next we choose one of the two edges of the triangle for which the particles have different colours, for example, the edge (j, l) in [figure 7.1](#). Since this edge has different colours at either end we know it crosses the interface and is therefore also a part of the interface triangulation.

To proceed we then check the circumcircles of all nearby particles except the third particle from the previous triangle, in this case particle k . When a circumcircle is found that contains no other particles we add the corresponding triangle to the triangulation. This step is repeated by each time choosing the single new edge which crosses the boundary. In this manner the interface triangulation is built up until we find either a wall boundary *or* the triangulation joins up with itself. After the interface is complete we check for new interface pairs which are not part of the current triangulation and, if any are found, repeat the algorithm for this pair. In this way the interfaces of separate droplets can be built up. Shown in [figure 7.2](#) is the output from the algorithm as applied to an actual SPH simulation.

The reason that the algorithm described here is $O(N)$ rather than $O(N \log N)$ is because we already know something about the structure of the points we are triangulating. For a given

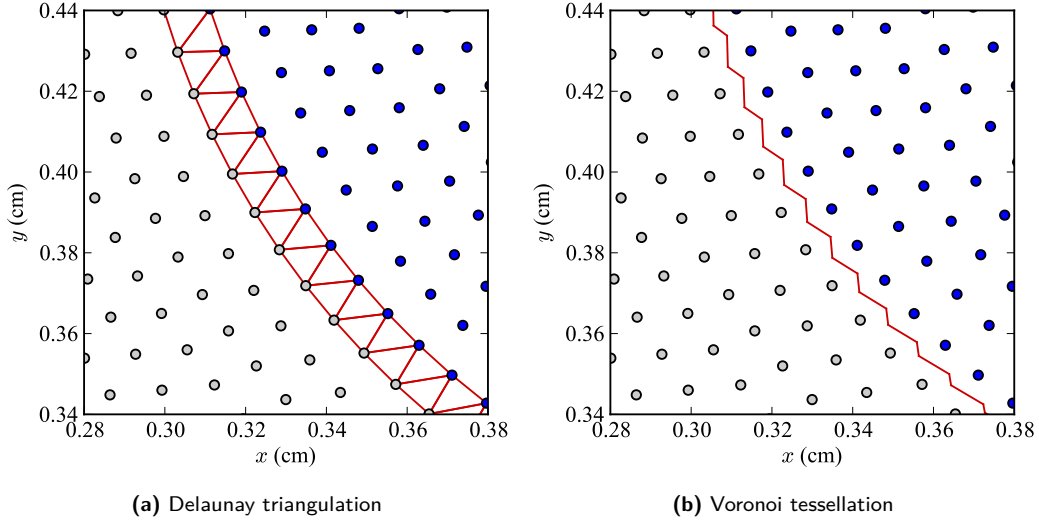


Figure 7.2: When applied to an interface the triangulation algorithm described correctly produces the Delaunay triangulation for the interface thereby allowing us access to the topology of the interface and the Voronoi tessellation.

edge the third point which forms the next Delaunay triangle will be within $2h$. This means we can use the linked cell grid to find both our potential triangulation points, and the other edges. This is a constraint which other more general algorithms do not have, hence the impact on their runtime speed.

7.2 Surface Tension

Once the structure of the interface has been computed one can use the knowledge of this to calculate the surface tension. For each Voronoi edge i we can take the unit vector $\hat{\mathbf{s}}_i^+$ which points from the midpoint of edge i to the mid point of the connected edge $i + 1$ and similarly, the unit vector $\hat{\mathbf{s}}_i^-$ points from the midpoint of i to the midpoint of $i - 1$. The sum of these two will be zero if the interface is flat. If we imagine the Voronoi edges to be a piecewise linear approximation of some smooth curve $\mathcal{C}$ then if the curvature κ of $\mathcal{C}$ is non zero at the edge i we have that $\mathbf{s}_i = \hat{\mathbf{s}}_i^+ + \hat{\mathbf{s}}_i^- \neq 0$. Moreover, $|\mathbf{s}_i| \approx -\kappa\delta\hat{\mathbf{n}}$, where δ is the average length of a Voronoi edge. Therefore at edge i we have

$$\mathbf{f}_{\text{s.t.}} = -\alpha\kappa\delta\hat{\mathbf{n}} \approx \alpha\mathbf{s}_i. \quad (7.1)$$

Note the factor of δ in the above equation implicitly accounts for the finite non-zero length of a Voronoi edge. As we increase the resolution there are more Voronoi edges for a given interface length but this is exactly compensated for by the corresponding linear reduction of $\mathbf{f}_{\text{s.t.}}$. The force can then be distributed to the SPH particles so that they feel the correct surface tension. There are two possible choices for how to do this. The first option is to notice that each Voronoi edge is associated with exactly two SPH particles which are necessarily of different colours. We

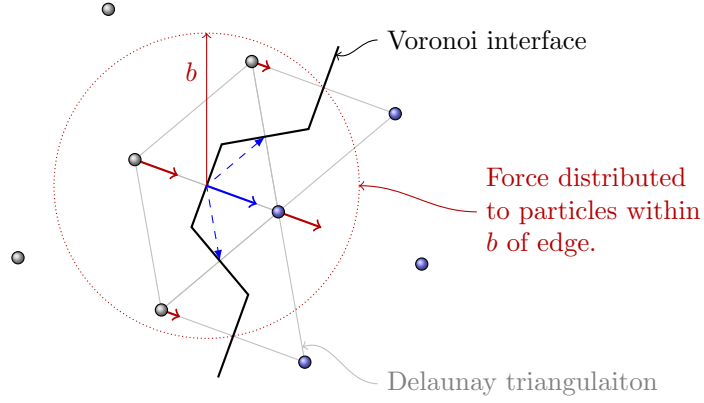


Figure 7.3: Based on the curvature of the interface the resulting surface tension force is distributed to nearby particles.

can therefore give each particle an acceleration equal to

$$\mathbf{a} = -\frac{1}{2} \frac{\alpha \mathbf{s}_i}{V \tilde{\rho}}, \quad (7.2)$$

where $\tilde{\rho}$ is the average density at this point of the interface;

$$\tilde{\rho} = \rho_1 f + \rho_2 (1 - f), \quad (7.3)$$

with

$$f = \frac{1}{2\pi} \arccos(\hat{\mathbf{s}}_i^+ \cdot \hat{\mathbf{s}}_i^-). \quad (7.4)$$

f is the ratio between the angle formed between the two vectors $\mathbf{s}_i^+$ and $\mathbf{s}_i^-$ and 2π thereby giving the fraction of a circle centred at the edge belonging to each phase. Applying an equal acceleration to the particles ensures that the particles from lighter phase are not accelerated towards or away from the interface quicker than particles from the heavier phase.

The second option for distributing the force is similar to the CSF model whereby the force is distributed according to a weighting factor to all nearby particles. As long as the weighting factor integrates to unity the overall force felt by the fluid will be correct therefore an obvious choice is to weight the force using the smoothing kernel. Of course we are free to choose the distribution length scale, b say, to be different from the smoothing length h thereby still allowing us some degree of sharpening of the surface forces as compared to the basic CSF model (cf. [section 6.1](#)). Thus for a each Voronoi edge i a particle j feels an acceleration given by

$$\mathbf{a}_j = -\frac{\alpha \mathbf{s}_i}{V \tilde{\rho}} W(|\mathbf{x}_i - \mathbf{x}_j|, b), \quad (7.5)$$

where $\mathbf{x}_i$ is the midpoint of the Voronoi edge i . Note that the acceleration here is still weighted by the average density $\tilde{\rho}$. In practice is found that distributing the resulting surface forces using the smoothing kernel with a distance $b = h/2$. gives a far better behaviour of the interface as compared to distributing the force only to the two Voronoi edge particles. The calculation of the vector $\mathbf{s}$ and the force distribution are illustrated in [figure 7.3](#).

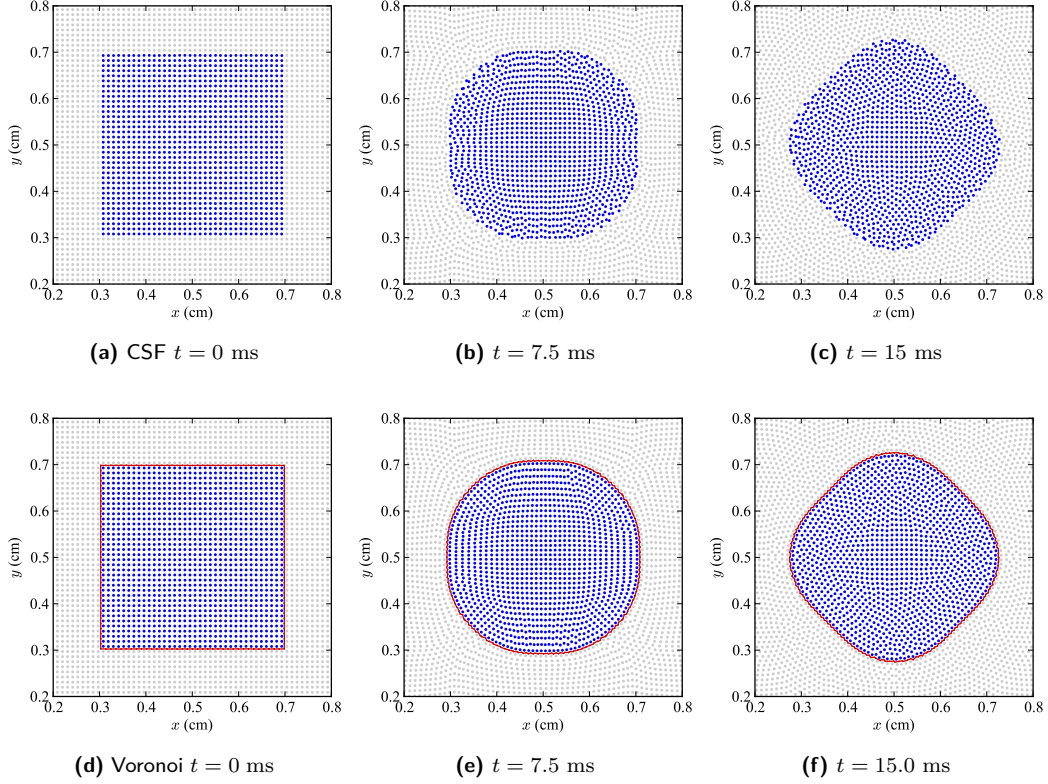


Figure 7.4: Simulation of the same droplet with identical parameters clearly shows how the new Voronoi based surface tension force displays improved particle positioning at the interface while correctly reproducing the dynamics of the droplet.

The surface tension formulation here was compared to the CSF surface tension formulation of [49] by simulating an initially square droplet with surface tension 0.025 N m^{-1} using 8.3×10^3 SPH particles. The SPH smoothing factor was $s = 1.3$ and both phases have equal viscosities (0.005 Pa s) and densities (1000 kg m^{-3}). The parameters for both simulations were identical. The particle positions are shown in figure 7.4 for both the CSF and Voronoi based simulations. It is clear that the particle positioning at the interface is far superior for the Voronoi surface tension as compared to the CSF. Although it is important to note this is the basic CSF model of [49] and not the corrected divergence model of [1], but even this still displays some spurious velocities.

The improved performance of the new method is still seen at higher density ratios. Figure 7.5 shows the final resting state of the initially square droplet where now the droplet has a density of 1000 kg m^{-3} and the outer fluid has a density of 10 kg m^{-3} giving a density ratio of 100. It is clear that in this case the Voronoi surface tension force still displays improved particle positioning relative to the CSF reference simulation. This improved particle position corresponds to lower spurious velocities. Both simulations correctly reproduce the expected Laplace pressure of 12.5 Pa .

Despite having to reconstruct the interface topology at each timestep the average time to

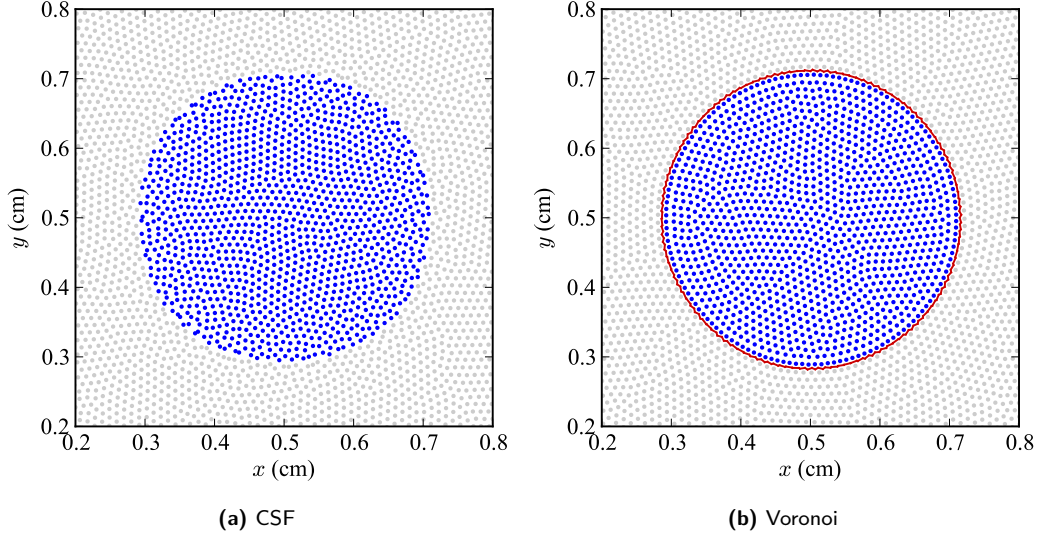


Figure 7.5: For the initially square droplet, with a density ratio of 100, both droplets assume the correct final shape and have the correct pressure jump of 12.5 Pa. However, once again the particle position with the Voronoi surface tension is much improved over the CSF reference simulation.

execute one iteration of the simulation was found to be consistently 20% lower for the Voronoi surface tension than the CSF surface tension throughout the course of the simulation. The reason for this is that when using the CSF model the interfacial stress (equation (3.51)), and its divergence, are calculated for every particle in the domain. These computations actually represent a significant fraction of the runtime for the SPH code. When using the Voronoi surface method, however, we use the Voronoi edges and distribute the surface tension forces to nearby particles only. This means that for particles more than $2h$ away from the boundary we do not have to perform any additional computations, thereby accelerating the code.

Accurately calculating the surface tension, and in principle contact, forces is one benefit of triangulating the interfaces in this fashion but knowing the precise interface allows us to improve the SPH simulations in another way too.

7.3 Pressure Discontinuity

At the surface of a gas bubble or liquid droplet there is a pressure discontinuity. To balance the inward surface tension the pressure is higher inside a droplet. This pressure difference is known as the Laplace pressure and is given by

$$\Delta P = \frac{\alpha}{R} \quad (7.6)$$

for a droplet of radius R . In SPH because the pressure force acts through the smoothed SPH gradient (equation (3.7)) the pressure at an interface does not exhibit a pressure discontinuity, instead it increases smoothly by ΔP over a width of $2h$. This smooth increase can be seen in

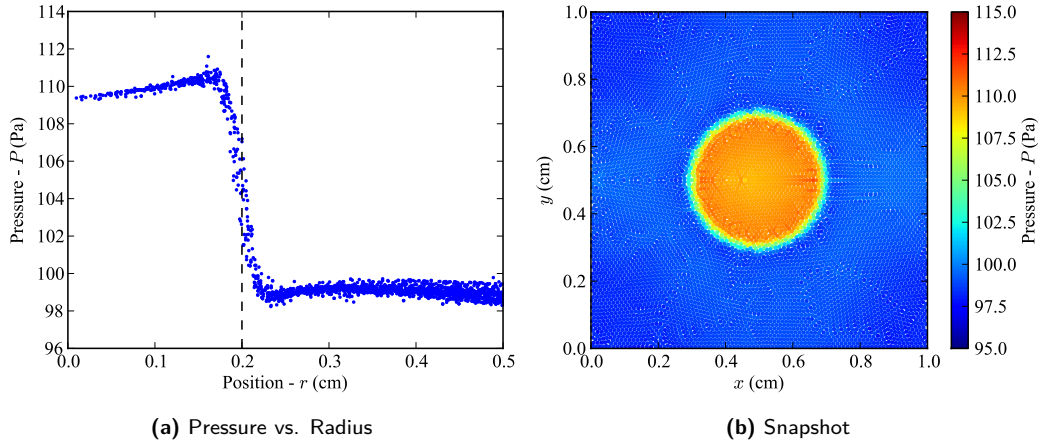


Figure 7.6: The Laplace pressure due to a curved interface in a droplet increases smoothly in SPH since the pressure acts over a width $2h$.

figure 7.6 which shows the output from a simulation of a droplet of radius $R = 2$ mm with a surface tension of $\alpha = 0.025$ N m⁻¹. This is true even when using the Voronoi surface tension as described above, due to the fact the pressure gradient is calculated by summing over all particles within a radius $2h$.

When using the Voronoi surface tension formulation, however, we have a well defined interface. Therefore we can handle the fact there is a pressure discontinuity in our system by treating the interface as if it were a boundary. To decouple the phases we use the boundary integral method detailed in chapter 5.

The basic idea is that the calculated interface is treated as if it were a solid wall at each timestep. For the pressure calculation the particles of each phase therefore see only other particles of their own phase and the lack of support is accounted for by calculating the kernel correction factor γ as if the interface represented a wall boundary. In this case there are no boundary particles but we can still calculate a pressure at the interface. At the interface i the values for σ and γ due to phase k are given by

$$\sigma_i^k = \sum_{j \in k} W_{ik}, \quad (7.7)$$

and

$$\gamma_i^k = \int_{\Omega_i^k} W(|\mathbf{x}_i - \mathbf{y}|) d\mathbf{y}. \quad (7.8)$$

This integral is calculated by using the direct numerical integral detailed in section 5.4.2.

Once we know σ and γ , the relative density at the interface is merely $\rho_i^{r,k} = V_0 * \sigma_i^k / \gamma_i^k$ which we can then use to calculate the pressure at the interface due to phase k by using the equation of state. There are now, however, two pressures at each interface edge. One from each phase, k . In general these two pressures will not be equal, i.e. there is a pressure discontinuity. The force on the interface due to this discontinuity is simply

$$\mathbf{f}_{p.d.} = -L\Delta P \hat{\mathbf{n}}_i,$$

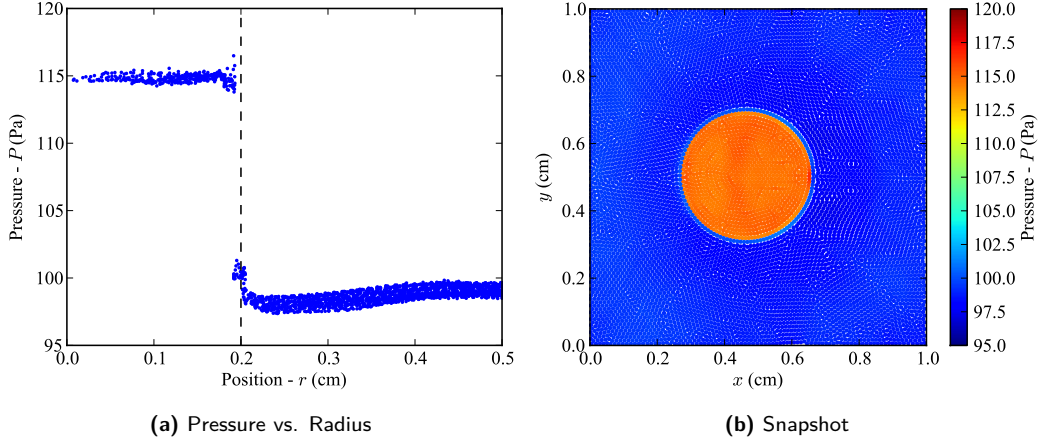


Figure 7.7: By treating the Voronoi interface with the boundary integral method we can create a true discontinuity in the pressure field as one would expect due to the Laplace pressure.

where L is the length of the interface edge. The force arising from this discontinuity is, as with the surface tension force, distributed to nearby particles using the smoothing kernel

$$\mathbf{a}_j = -\frac{L\Delta P \hat{\mathbf{n}}_i}{V\tilde{\rho}} W(|\mathbf{x}_i - \mathbf{x}_j|, b). \quad (7.9)$$

Because the pressure discontinuity force is distributed in the same way as the surface tension force we are actually distributing a net force given by

$$\mathbf{f}_i = -\alpha \mathbf{s}_i - L\Delta P \hat{\mathbf{n}}_i. \quad (7.10)$$

At equilibrium this net force should be zero. This means that for our simulations as time moves on we are distributing only small forces onto the interface particles, whereas for the CSF model a non-zero force is given to the interface particles even at equilibrium.

It is important to note that using this formulation, even though the pressure can be discontinuous the two phases still interact via pressure. Partly through the application of [equation \(7.9\)](#) but also through spacing. If the particles from each phase approach each other they move closer to the Voronoi interface, which is always in the middle of the interface particles. They then see this as approaching a wall, boundary which exerts a repulsive force through the pressure gradient [equation \(5.51\)](#). The viscosity across the phase boundary is calculated as it normally is using [equation \(3.32\)](#), i.e. the particles of both phases interact through the shear stress. There is, however, potential to add additional effects such as Marangoni induced interface rigidity.

By treating the the reconstructed interface as a boundary in this way, we are able to introduce a true discontinuity into the pressure field. The same simulation as was shown previously was run using this formulation with exactly the same parameters. One can see from [figure 7.7](#) that the pressure exhibits the expected discontinuity at the droplet interface. An additional bonus is that the particle spacing at the interface is maintained very nicely by the Voronoi surface tension and

there are no spurious velocities. However, the pressure jump ΔP is somewhat over estimated by this method. It seems that, due to the treatment of the interface using the boundary integral method, the simulations are very sensitive to the smoothing factor $s = h/\Delta x$. This could be ameliorated by subdividing the Voronoi edges into shorter segments, thereby improving the discrete approximation for the surface integral in [equation \(5.51\)](#).

A big potential benefit of handling the surface tension in this manner is that, because we explicitly account for the pressure difference on either side of the boundary, one can also account for differing resting pressures of the fluids. This would remove the pressure matching requirement discussed in [section 3.5](#) and allow each fluid to have a different equation of state. The advantage of this would be that the air phase in our unsaturated simulations would no longer have to have a higher numerical speed of sound (and hence lower compressibility) than the water phase. This means that, for simulations which are not limited by the capillary timestep, the acoustic timestep could be larger. For air-water systems this difference could be up to a factor of 30 depending on the resolution and surface tension.

7.4 Conclusions

While the work shown in this chapter is work-in-progress, a good basis for future developments has been achieved. The behaviour of the Voronoi surface tension appears to be superior to the CSF surface tension with regards to the spurious velocities. Further, having a well defined interface allows us to make additional improvements to the method. For example, accounting explicitly for pressure discontinuities and allowing each phase to be governed by a different equation of state. This allows an increase in the acoustic timestep for some unsaturated simulations.

The improved performance of this method also suggests that the simulation of triple points should be enhanced through the reduction of spurious velocities. Additionally, due to knowing the angles any droplets make when in contact with a wall boundary, there is the possibility to include contact angle hysteresis. Although both of these points remain to be investigated.

Chapter 8

Mass Transport in Packed Beds

The flow of liquids through porous media occurs in many systems, both industrial and natural, and has implications for many fields ranging from hydrology to petroleum engineering. Many studies which examine porous media and analogous systems measure the dispersion caused by the liquid motion through the pore or interstitial space. The dispersion, the spread of any solutes, is a combination of both molecular diffusion and hydrodynamic effects. Typically large scale studies consider the overall dispersion to be diffusive, or ‘Fickian’. That is not to say that only diffusion is important, but that the combined action of the flow and molecular diffusion can still be described by the same equations as purely molecular diffusion, albeit with a much larger constant of proportionality - D the dispersion coefficient.

Porous media have a wide variety of properties with pores ranging from nanometres up to the sorts of scales encountered in heap leaching where the interstitial spaces can be $\sim$ centimetres, as well as varying structures due to the formation processes, such as dead-end pores and inaccessible regions. Even for porous media on the same scales, different studies often produce different values for the measured dispersion coefficients depending on the exact conditions used [44, 45, 53]. Additionally, recent evidence suggests that such advection-diffusion type models are not valid ways of scaling up the microscopic dynamics to macroscopic continua models. Several papers have approached the problem from a theoretical perspective and provide mechanistic justifications for this viewpoint [96–99]. Others have demonstrated that deviation from advection-diffusion type models can be seen experimentally in both saturated and unsaturated flows [43]. In this chapter we investigate the behaviour of unsaturated flows in packed beds. To do this we utilise the insights from the previous chapters about simulating multiphase systems with SPH.

8.1 Dispersion Mechanisms

For fluid flowing through a packed bed there are two physical effects which act to spread any chemical species being transported in solution; hydrodynamic dispersion and molecular diffusion. The former is caused by the action of the fluid flow as the name suggests. As fluid flows around an obstacle the path-lines can split and then be merged with other incoming path-lines thereby spreading solutes in a mixing process somewhat akin to a random walk. This probabilistic viewpoint has lead to several studies which attempt to model these deterministic, albeit complex, processes as stochastic processes on simplified or idealised networks [30]. Molecular diffusion takes place at a very different scale and is caused by the random thermal motions of the fluid’s constituent molecules acting on the molecules of dissolved chemicals.

As is widely known, for a substance, which is diffusing according to the diffusion (or heat)

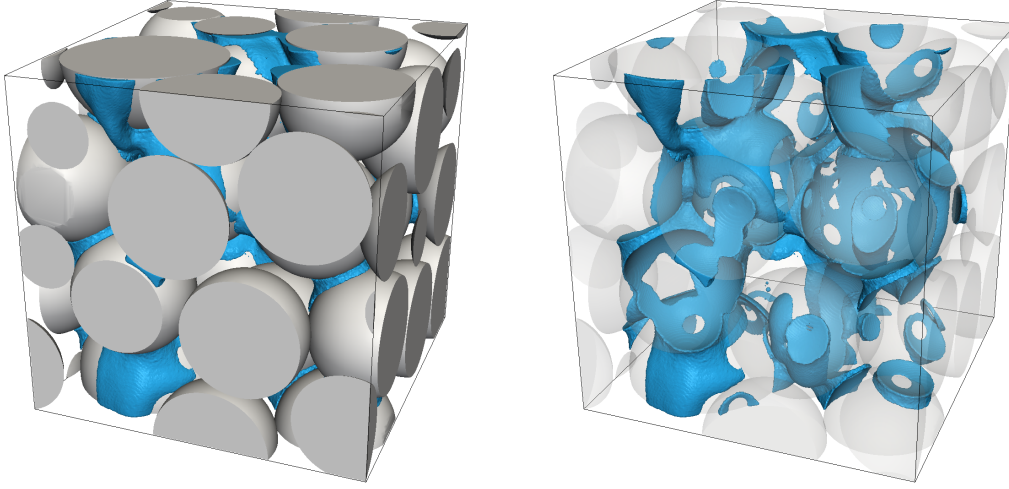


Figure 8.1: Example output from a 3D periodic packing of bi-disperse spheres with domain size $(4.22 \text{ cm})^3$.

equation,

$$\frac{\partial C}{\partial t} = D_m \nabla^2 C,$$

the variance of the distribution of C increases linearly with time. Thus one finds that $D_m = \text{VAR}(C)/2t$ is a well defined constant, called the diffusion coefficient, which specifies how quickly the substance spreads due to diffusion. By drawing an analogy with diffusion we can specify the *hydrodynamic dispersion coefficient*

$$D_h = \frac{\text{VAR}(C)}{2t},$$

which captures the rate of spreading caused by any fluid motion in the system.

The relative importance of the molecular diffusion compared to advective transport is captured by the dimensionless diffusion Péclet number

$$\text{Pe}_m = \frac{UL}{D_m}. \quad (8.1)$$

It is the ratio of the advective transport (the product of velocity U and a length scale L) to the molecular diffusion coefficient D_m . This is very similar to the dispersion Péclet number which is the ratio of the advective transport to the total dispersion arising from both molecular and hydrodynamic sources;

$$\text{Pe}_d = \frac{UL}{D_m + D_h}. \quad (8.2)$$

For the systems simulated in this thesis we find that $\text{Pe}_m \approx 10^6$; such a high diffusion Péclet number indicates that, for the interstitial flow, molecular diffusivity plays a negligible role and we would expect the hydrodynamic effects to completely dominate. This is analysed in detail

in [section 8.3](#).

8.1.1 Péclet Numbers & Dispersivities

In order to compare different systems it is convenient to introduce dimensionless versions of the quantities of interest. This is done by considering characteristic properties for each system. For instance, each system has a characteristic length scale L and a characteristic velocity U . The characteristic length scale is the size of the particles (typically ~ 1 cm) and the characteristic velocity is the ‘seepage velocity’, which is discussed in the next section. By combining these quantities one can write the dimensionless time

$$\tilde{t} = \frac{tU}{L}. \quad (8.3)$$

In a similar fashion one can define the dimensionless dispersion coefficient

$$\tilde{D} = \frac{D}{UL} = \frac{1}{\text{Pe}_d}. \quad (8.4)$$

$\tilde{D}$ is the inverse of the dispersion Péclet number (see [section 8.1](#)) and is one of the quantities, which is often reported by experimental studies into dispersion [32]. These dimensionless quantities let us compare the behaviour between different systems.

If one has two systems in which the seepage velocities are the same but, which have different typical length scales, for instance, then it will take fluid in the system with the larger characteristic length longer to explore its space of possible configurations. Therefore comparing the systems at the same real time is not very meaningful. If two systems have different scales but the rescaled quantities display the same trends (i.e. they collapse onto a single curve) then we can infer that the mechanisms governing the behaviour are, in principle, the same. Vice versa, different underlying mechanisms will manifest themselves as curves which do not collapse under such rescaling.

In the definition of the diffusion tensor in [equation \(2.6\)](#), the magnitude of the dispersion is written as a product of a so-called dispersivity - α - and the flow velocity $|\mathbf{u}|$. Thus for the Fickian type dispersion described by [equation \(2.5\)](#) the amount of dispersion is related to the Péclet number by

$$\text{Pe}_d = \frac{L}{\alpha}, \quad (8.5)$$

meaning the dispersivity is given by the ratio of the characteristic length scale to the dispersion Péclet number.

8.1.2 Seepage Velocity

All of the simulations in this thesis start with zero initial velocity throughout the system. The fluid is then accelerated by gravity meaning its speed will increase with time. After some time, however, the water in our simulations will have reached a point where (on average) the

acceleration due to gravity will be canceled by the action of viscosity and capillary interactions with the packed bed. This means that at long times the water will move with a constant mean velocity through the system. This mean velocity is called the seepage velocity.

The time evolution of the mean fluid velocity for an idealized system is given by¹

$$v_s = \frac{\rho \epsilon \theta_s g}{\mu L \tilde{c}_d} \left(1 - \exp \left(- \frac{\mu L \tilde{c}_d}{\rho \epsilon \theta_s} t \right) \right), \quad (8.6)$$

where L is a characteristic length scale, ϵ is the porosity, θ_s is the saturation and $\tilde{c}_d$ is the drag coefficient which depends on the system's geometry and possibly θ_s . Thus for this ideal system

$$\lim_{t \rightarrow \infty} v_s = \frac{\rho \epsilon \theta_s g}{\mu L \tilde{c}_d}. \quad (8.7)$$

By measuring the actual seepage velocity seen in our simulations we can then use this equation to measure properties of the system. The measured seepage velocity is simply the mean velocity of the water in the system, i.e.

$$v_s(t) = \frac{\int_{\text{water}} \mathbf{u}(t) \cdot \hat{\mathbf{g}} dV}{\int_{\text{water}} dV}. \quad (8.8)$$

Knowing v_s also allows us to rescale the measured dispersion coefficients as discussed above.

In our simulations the domain is periodic in all directions, thus we are simulating a closed system. In reality heaps have a finite size and the water (leaching solution) is irrigated into the heap from the top at a specified rate. This rate is given by the *superficial* velocity which is related to the seepage velocity by $v_{\text{superficial}} = v_s \epsilon \theta_s$. In reality, therefore, one is specifying not the saturation but the superficial velocity. Thus it is worth noting that our simulations are in effect backwards from experimental studies since we specify the saturation and observe, depending on this, a given (seepage) velocity. Because of the relationship between $v_{\text{superficial}}$ and θ_s for experimental studies, we can also access higher saturations with our simulations.

8.2 Measuring Dispersion

In this thesis the primary concern for the application of SPH to packed beds is how fluid transport is affected by various parameters. As discussed above, the hydrodynamic dispersion coefficient is typically what is measured when studying transport phenomena.

Because the distance by which each fluid parcel has moved in a certain time is usually given the symbol ζ in the literature, we will denote the displacement vector between times t and $t + \tau$ as $\zeta_t(\tau)$ in general, and for a specific SPH particle i as $\zeta_{t,i}(\tau)$. The dispersion coefficient in the α -coordinate is then given by

$$D_\alpha = \frac{\text{VAR}(\zeta_t^\alpha(\tau))}{2\tau}, \quad (8.9)$$

¹See [appendix A.6](#) for the derivation.

where the variance is calculated over different points in the fluid. A more general form for equation (8.9) is

$$D_{\alpha\beta} = \frac{\text{Cov}(\zeta_t^\alpha(\tau), \zeta_t^\beta(\tau))}{2\tau}. \quad (8.10)$$

This is the dispersion *tensor*, which not only describes spread along each direction but how they are correlated with one another. The diagonals of the dispersion tensor are the normal dispersion coefficients, $D_{\alpha\alpha} \equiv D_\alpha$. The displacement of a particle in a time interval τ , depends on the time t from which we start measuring. If we were simply to take $t = 0$ and calculate the displacement since the start of the simulation, the measured dispersion would be heavily dependent on the particular flow seen. To counter this effect, for a particular time interval τ , we divide the total simulation time T into $N = \lfloor T/\tau \rfloor$ segments and calculate the total dispersion over these. Thus the dispersion tensor is numerically calculated by

$$D_{\alpha\beta}(\tau) = \frac{1}{2\tau} \text{Cov}(\Xi^\alpha, \Xi^\beta), \quad (8.11)$$

where

$$\Xi^\alpha = \bigcup_{n=0}^{N-1} \{\zeta_{n\cdot\tau, i}^\alpha(\tau)\}.$$

In words, Ξ is the set, over all particles, of displacements after each interval of length τ over the course of the simulation. The dispersion parallel to the direction of flow is known as the axial, or longitudinal, dispersion whereas the dispersion perpendicular to the flow is known as radial, or transverse, dispersion. Thus there are two different dispersion coefficients to consider;

$$D_{\text{axial}} = \hat{\mathbf{u}} \cdot \mathbf{D} \cdot \hat{\mathbf{u}}, \quad D_{\text{radial}} = \hat{\mathbf{u}}_\perp \cdot \mathbf{D} \cdot \hat{\mathbf{u}}_\perp, \quad (8.12)$$

where $\hat{\mathbf{u}}_\perp$ is a unit vector in a direction perpendicular to the velocity $\mathbf{u}$, as discussed below the exact direction of $\hat{\mathbf{u}}_\perp$ is only important for anisotropic media. Since our flow is driven by gravity, the average flow is perpendicular to the vector $\mathbf{g}$. If gravity points in the negative y direction, say, then we have

$$D_{\text{axial}} = D_{yy} = \frac{\text{VAR}(\zeta^y(\tau))}{2\tau}, \quad (8.13)$$

and

$$D_{\text{radial}} = D_{xx} = \frac{\text{VAR}(\zeta^x(\tau))}{2\tau}. \quad (8.14)$$

In three dimensions it is possible to have $D_{xx} \neq D_{zz}$ (the two radial directions) if there is anisotropy between these directions. This, however, is not the case for any of the simulations in this thesis.

As τ increases the number of segments of length τ that fit into the simulated time T necessarily decreases. Thus for our simulations the estimates for D at higher τ have fewer samples and are subsequently more sensitive to the history of the flow seen in the simulation.

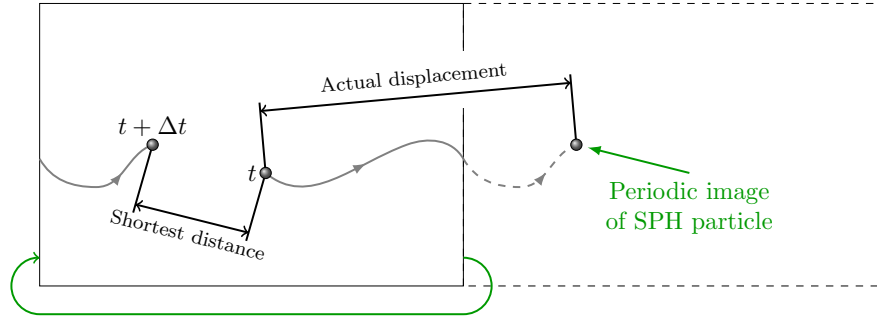


Figure 8.2: Taking the both the simple distance and the periodic distance between a particle's position at time t and a later time $t + \Delta t$ gives the wrong distance travelled. Instead the displacements are integrated in time to give the correct displacement since this is blind to the periodic boundaries.

8.2.1 Propagators

The dispersion coefficient is a useful quantifier because, with it, one can draw a direct analogy with molecular diffusion and it can be fed directly into advection-dispersion type models. However, it is in some sense an aggregate measure; by considering only the variance of ζ we are losing information about the flow (two different distributions can have the same variance). In the porous media literature another way of characterizing the spreading properties of flow is common; namely the so-called propagator.

The propagator is the probability density function of seeing a particular rescaled displacement $\zeta / \langle \zeta \rangle := \hat{\zeta}$ after some time τ ; $p(\hat{\zeta})$. Because of this rescaling the propagator only exists for the axial flow direction because in all other orthogonal flow directions we have $\langle \zeta \rangle \equiv 0$.

The displacement is rescaled so that for all times $t \neq 0$ the mean of the propagator is unity. For $t = 0$ the rescaling operation is ill defined because $\langle \zeta \rangle = 0$. The behaviour of D at long times is reflected in the limit of the propagator. If the flow induces a diffusive (Fickian) type spreading then we have

$$\lim_{t \rightarrow \infty} p(\hat{\zeta}) = \delta(\hat{\zeta} - 1),$$

i.e. $\lim_{t \rightarrow \infty} \text{VAR}(\hat{\zeta}) = 0$ and at large times p will be well approximated by a Gaussian with $\sigma^2 = \frac{D}{v^2 t}$ and $\mu = 1$.

8.2.2 Path-lines

Calculation of both the dispersion coefficient(s) and the propagator relies upon knowing the displacements $\zeta_{t,i}(\tau)$ for each particle i , but due to the periodic domain they cannot simply be calculated using the usual distance, $(\mathbf{x}(t) - \mathbf{x}(t + \tau))$, or even the periodic distance, $\mathbf{d}\mathbf{x}(\mathbf{x}(t), \mathbf{x}(t + \tau))$. If a particle traverses more than half a period this distance will be wrong, as is illustrated in figure 8.2. To overcome this the displacements ζ could be calculated by

integrating the velocities

$$\zeta_{t,i}(\tau) = \int_t^{t+\tau} \mathbf{u}_i(t') dt'.$$

This equation gives the correct displacement because the velocities are not affected when crossing a period. The interval with which output is written by the simulation is somewhat larger than the actual timestep used for integrating the equations of motion. As such the velocities actually vary slightly over the course of time between two output files and for this reason we cannot actually use the velocities seen in a given output file to integrate for position without accumulating numerical errors. Instead we simply calculate the periodic distance between two output steps and add this to the total displacement, giving

$$\zeta_{t,i}(\tau) = \sum_{t'=t}^{\tau} \mathbf{dxp}(\mathbf{x}_i^{(t')}, \mathbf{x}_i^{(t'+\Delta t)}), \quad (8.15)$$

where $\cdot^{(t)}$ indicates the value at timestep t , and Δt is the output interval (not the integration timestep used for the simulation, cf. [section 3.4](#)). [Equation \(8.15\)](#), however, still relies on the fact that in Δt the particles have moved a distance less than half the period; a criterion which is easily satisfied for the values of Δt used in this thesis.

By integrating the fluid paths in this fashion our calculations of the displacements are not limited by the size of our systems and we can consider our fluid to be moving through an infinite (periodic) material. It is important to note however, that even though our fluid paths are blind to the periods of our system this does not mean the fluid is not affected by the periodicity of the domain *and* the flow within it. It is well known that for steady state flow in a regular periodic domain, the size of the system can have a large effect on the results [\[71, 105\]](#). This is because for periodic system which display regular symmetries, if the flow direction is aligned to one of these axes then the stream-lines are closed and a particle following a particular stream-line will always retrace the same route through the domain.

In [Zhu & Fox \[105\]](#) they argue that if the flow direction is not parallel to an axes of symmetry then the flow paths are necessarily non repeating. This is not necessarily true; depending on the angle and the system sizes it is possible that the stream-line will reconnect and the flow will truly repeat after some (possibly large) number of domain traversals. Further, despite using an irregular periodic packing [Lowe & Frenkel \[71\]](#) saw a dependence on the system size. This is because flow structures can exist for which the extend up to the size of the system.

For simulations in which the flow is not steady-state, the stream-lines cannot be closed. Most simulations presented in this thesis were unsaturated and unsteady, combined with the fact that some of the domains were irregular packings means that periodicity is not expected to play a large role in the results obtained. For unsteady flow the stream-line is only defined instantaneously and for unsaturated flow may not be defined at all for a particular fluid phase, in both situations it is better to consider the *path-line* of a fluid particle; indeed the path-line is exactly what we are calculating with [equation \(8.15\)](#). Example path lines through the extended periodic domain are shown in [figure 8.3](#) for a 2D simulation of irregularly packed circular particles. The black square indicates the extent of the periodic domain.

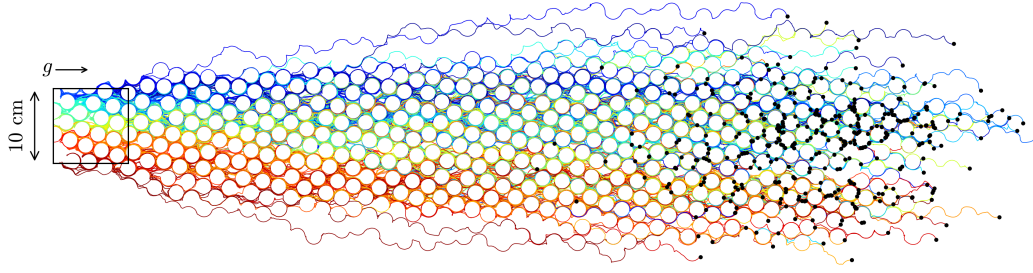


Figure 8.3: The integrated flow paths of 400 SPH particles through a periodic irregularly packed domain. The black box indicates the extent of one periodic domain and the black dots show the locations of the SPH particles after $t = 6$ s.

Because of the periodic nature of the simulation domains we are in effect simulating part of an infinite periodic domain which is, in some sense, uniformly saturated throughout. The simulation domain is therefore a *representative elementary volume* of a larger system at stationary (not steady) state. This means the simulations cannot capture, for instance, the spread of a saturation front moving through the heap after initial set-up but do capture the transport of dissolved solutes in a running heap.

8.2.3 Ergodicity & Averaging

A system is said to be *ergodic* if the amount of time a system spends in a particular macrostate is proportional the volume of its configuration space, which is equivalent to the macrostate. This means if one were to watch such a system for a long time (compared to the correlation time of the system) then any time averages calculated are equivalent to ensemble averages. In the context of our simulations, the ensemble average does not equate to averaging over the fluid within the domain, rather, averaging over repeats of a simulation. The systems studied in this thesis are usually *not* ergodic. There are large hysteresis effects seen in systems such as this with strong capillarity, and this necessarily breaks ergodicity.

All averages in this chapter are calculated as time averages except where explicitly stated (for instance when calculating $v_s(t)$). This means that any statistics reflect only the portion of configuration space available to us from the initial conditions used. We will see in [section 8.5.2](#) that this can have an effect on the long term observed properties. However, there are good reasons for considering time averages. Because we are interested in the real behaviour we would see in a heap we are actually interested only in those states reachable by our system. In a real heap fluid is continually added meaning that eventually the fluid will begin to flow and so we are less interested in the configurations where the fluid is entirely held up.

The time averaging of quantities is only begun after a specified time has elapsed. This allows the system to reach stationarity before averaging commences. Stationarity does not mean that the system is in a steady-state where nothing is varying with time. Instead it means that the combination of gravity and viscosity have driven the system into the region of configuration

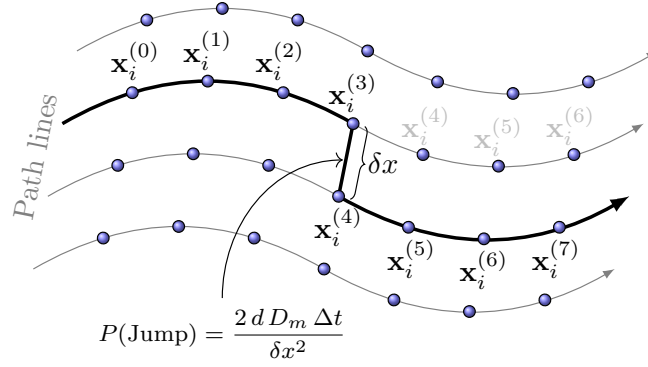


Figure 8.4: The path lines followed by each fluid parcel are updated with a probability proportional to the molecular diffusion D_m at each timestep, thereby simulating the action of molecular diffusion.

space in which it will then stay indefinitely. This means therefore that any time averages we make will be meaningful of the configurations reachable by the system.

8.3 Simulating Molecular Diffusion

To test whether, molecular diffusion is indeed negligible, a comparison simulation was run which accounted for molecular diffusion by way of a continuous time random walk. The SPH simulations themselves do not consider molecular diffusion. This is because to measure the hydrodynamic dispersion we consider only the path lines of SPH fluid particles. Despite this, molecular diffusion can be simulated *a posteriori* using the output of the SPH simulations. This works by swapping which path lines tracer parcels are following with a probability that leads to the expected molecular diffusion. If a tracer particle is having its pathline calculated according to the positions of an SPH particle i then at a given timestep there is a probability

$$P(i \rightarrow j) = \frac{2dD_m\Delta t}{\|\mathbf{x}_{ij}\|^2}, \quad (8.16)$$

of jumping to a neighbouring SPH particle j and using its positions (until another jump occurs). This means that it is possible that we can end up with two (or more) tracers which follow the same SPH particle for some time. This process is illustrated in figure 8.4, the integration of the path lines as given by equation (8.15) is modified so that we do not always take position $\mathbf{x}_i$ for pathline i , the pseudo code below shows how the calculation of $\zeta_i^{(t)}$ is different in this case.

The neighbouring particles chosen as candidates to jump to include only fluid particles of the same colour as i , and wall particles. Further, i must not be included in the list of potential jump candidates, otherwise $P(i \rightarrow j)$ (and hence the behaviour of the SPH code) is undefined. In reality, if floating point exceptions (FPEs) are enabled the code will exit because of a divide-by-zero exception. Otherwise, if FPEs are not switched on, the code will most likely run but the observed diffusivity will be somewhat lower than D_m . This effect is especially pronounced

Algorithm 1 Path Line Calculation with Diffusion

```

for  $i \in \mathcal{F}$  do                                     ▷ Initialize indices and displacements.
     $a_i \leftarrow i$ 
     $\zeta_i^{(0)} \leftarrow (0, 0, 0)$ 

    for  $t \in \{0, \Delta t, 2\Delta t, \dots, T - \Delta t\}$  do           ▷ Loop over output timesteps.
        for  $i \in \mathcal{F}$  do
             $N_i \leftarrow$  Particles within  $2h$  of  $\mathbf{x}_i$            ▷ Neighbours of  $i$ 
             $j \leftarrow$  Random element of  $N_i \setminus \{i\}$        ▷ Chosen with uniform probability.
             $p \leftarrow 2dD_m\Delta t / |\mathbf{d}\mathbf{x}(\mathbf{x}_j, \mathbf{x}_i)|^2$      ▷ Jump probability.

            if  $U[0, 1) < p$  then
                 $b_i \leftarrow j$ 
            else
                 $b_i \leftarrow a_i$ 

             $\zeta_i^{(t+\Delta t)} \leftarrow \zeta_i^{(t)} + \mathbf{d}\mathbf{x}(\mathbf{x}_{b_i}^{(t+\Delta t)}, \mathbf{x}_{a_i}^{(t)})$ 
             $a_i \leftarrow b_i$                                ▷ Store path line index for next iteration.

```

for lower s , since it is more likely that i is chosen in this case.

The fact that we are less likely to move to a pathline which is farther away ($P(i \rightarrow j) \propto^{-1} \delta x^2$) is compensated for by the fact that the pathline is, in fact, farther away. This implementation of a CTRW is different from the usual method used for saturated flows [12–14] in which the jump size and direction are completely random. The advantage this method provides is that for our unsaturated flows the diffusing tracers necessarily follow the correct phase at all times (by restricting the jumps to SPH particles of the same phase). This difference in implementation, however, means the scheme should be verified by comparing its results to systems with known behaviours.

8.3.1 Timestep Constraint

Simulating diffusion as a random walk in this way places another constraint on the timestepping scheme used. The probability of swapping must not exceed unity. Taking $P(i \rightarrow j) \leq 1$ and rearranging implies that our timestep

$$\Delta t \leq \frac{\Delta x^2}{2dD_m}. \quad (8.17)$$

This constraint on the timestep is not due to stability, as is the case for the constraints discussed in [section 3.4.1](#). Instead, if our timestep exceeds this then our probability of swapping could become larger than 1 and the resulting diffusivity would not be D_m but some smaller value.

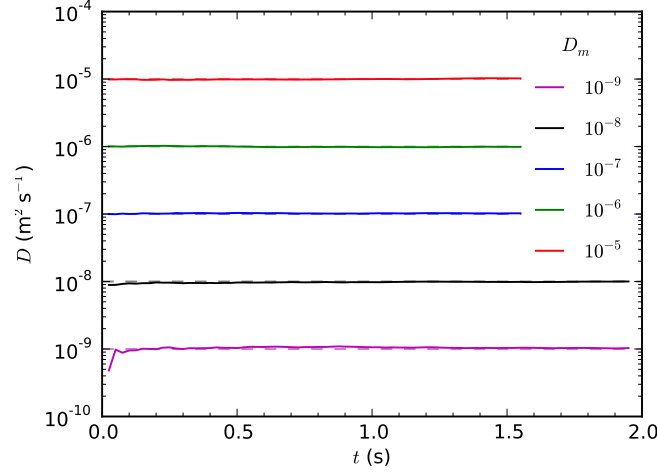


Figure 8.5: Simulating molecular diffusion in a 2D quiescent fluid gives us back the expected diffusion coefficient for a wide range of values of D_m ($\text{m}^2 \text{s}^{-1}$).

8.3.2 Validation & Error Analysis

This scheme was confirmed by simulating diffusion on top of a stationary velocity field with a given D_m and confirming that

$$\lim_{\tau \rightarrow \infty} \frac{\langle \zeta_{0,i}^2(\tau) \rangle}{2\tau d} = D_m. \quad (8.18)$$

The simulation domain was a $(1 \text{ cm})^2$ periodic domain filled with quiescent water, the simulation used 1×10^4 SPH particles. A range of molecular diffusion coefficients were simulated, each using 5000 tracer particles. The results are shown in [figure 8.5](#). We would expect large relative error at small times due to the finite step sizes. This is an effect of the finite size of the step sizes. Since the tracers jump only to neighbouring particles, our step size here is always $\approx \Delta x$ meaning that we are simulating a (lazy) *random walk* and not a true Brownian motion. It is easy to show that for such a random walk the measured diffusion coefficient will tend to that of a true Brownian motion as t tends to infinity and that the relative error associated with this convergence is given by $\varepsilon = \Delta t/t$. This error at small times is less pronounced for higher resolution simulations because as Δx decreases so too does Δt . Assuming the diffusion timestep as given by [equation \(8.17\)](#) is the limiting criterion then

$$\varepsilon = \frac{\Delta x^2}{2dD_m} \frac{1}{t}$$

thus doubling the resolution will quarter the error associated with the finite step size.

For the flows through packed beds D_m was set to $1.5 \times 10^{-9} \text{ m}^2 \text{s}^{-1}$ (for the water phase), to match the diffusion of salt in water at room temperature, but no difference was seen in the observed dispersivities. Another reason that in our systems molecular diffusion is unlikely to play a significant role (even for relatively high D_m) is that the flow is unsaturated. Even for very high D_m particles cannot diffuse onto path lines which are more than $2h$ away this means that diffusion between separate droplets is not possible, and the motion of the droplets is the

only transport mechanism available.

8.4 Taylor Dispersion

Apart from the joining and splitting of nearby pathlines, differences in velocity also cause dispersion. Shear flow acts to spread a substance being transported because the flow nearer the walls is slower than further from the walls. If the substance has $D_m = 0$ and the pathlines are closed² then the theoretical dispersion coefficient in the direction of flow (axial) does not actually exist. Because the pathlines are closed we can validly say that a fluid parcel on a given pathline moves with some average velocity $\langle \mathbf{u} \rangle$, meaning that the distance moved in any given direction after time t is $\zeta_\alpha \approx \langle u_\alpha \rangle t$. The relative error with this approximation decreases as t increases. Thus our dispersion coefficient is given by

$$\begin{aligned} D_\alpha &= \frac{\text{VAR} \left(\int_0^t u_\alpha(\tau) d\tau \right)}{2t}, \\ &\approx \frac{\text{VAR}(\langle u_\alpha \rangle t)}{2t}, \\ &\approx \frac{1}{2} t \text{VAR}(u_\alpha) \end{aligned} \quad (8.19)$$

Therefore D_α will either increase without bound linearly with time or be zero (if $\text{VAR}(\langle u_\alpha \rangle) = 0$). Since it takes of the order of 1 dimensionless time unit $\tau = 1$ for a fluid parcel on a closed pathline to sample all of its velocities we would expect deviation from the long time limits for $\tau \lesssim 1$.

In systems with $D_m > 0$ any solutes being transported on a particular pathline have a chance to move to a neighbouring pathline thereby making the solute's path no longer closed. This means that eventually there should be a well defined $\mathbf{D}$ but the time required to reach this can be large. For small D_m the shear flow in the pipe acts to enhance the dispersion (for the reasons described above), but for higher D_m the diffusion can cause a suppression of the dispersion. The reason for this is that as D_m is increased the particles diffuse through the cross-section and thus experience velocities from more of the path lines. This has the effect of making each tracer particle's average velocity closer to the mean velocity in the system. Thus the tracers are spread less by the flow.

8.4.1 Flow in a Pipe

The fact that in the absence of D_m and with closed pathlines we have $D_\alpha \sim t$ allows us to check that there is no undue numerical diffusion in the SPH method. 2D Poiseuille flow was simulated for a pipe with width $L = 1$ cm and fluid having density $\rho = 1000$ kg and viscosity $\mu = 0.001$ Pa s. The driving force was chosen so that the maximum velocity in the middle of the pipe is $v_{\max} = 1.75$ cm s⁻¹. This corresponds to a Reynold's number of 175. The simulation

²I.e. $\forall \mathbf{x} \in \Omega, t > 0 \exists T > 0 : \mathbf{x}(t + nT) = \mathbf{x}(T)$

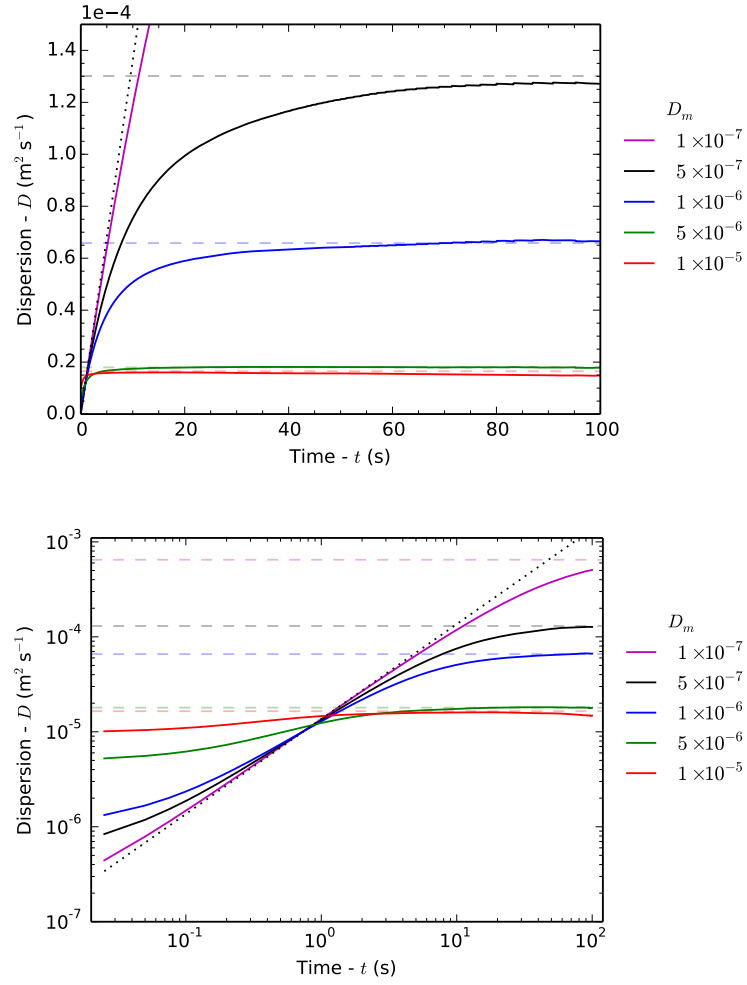


Figure 8.6: Simulating molecular diffusivity on top of flow in a 2D pipe (Poiseuille flow) shows that for low D_m we see an enhancement of mixing due to shear flow. The asymptotic limit of the measured dispersion agrees well with the theoretical values (dashed). The lower plot is the same as the upper but with logarithmic axes.

was run for 100 s which is long enough for all but the lowest D_m to approach their asymptotic values.

For the case with $D_m = 0$, [equation \(8.19\)](#) tells us that the expected Taylor dispersion in this case is $D(t) = \frac{2}{45} v_{\max}^2 t$. The measured value for axial dispersion exponent is 0.99996 ± 0.00001 ; a value differing by only 4 parts in 100,000 from the expected analytical value. The reason for this excellent agreement is because the laminae of SPH particles flow past one another perfectly with no lateral movement.

Next a range of molecular diffusions were simulated on top of this flow. The resulting dispersion coefficients as a function of time are shown in [figure 8.6](#). The deviation from the expected purely Taylor dispersion (dotted line) is caused by the tracer particles sampling different velocities over

time. The theoretical asymptotic value of the dispersion D is

$$D = D_m + \frac{v_s^2 L^2}{210 D_m}, \quad (8.20)$$

where v_s is the usual seepage velocity and L is the pipe width. For our D_m s the asymptotic values have been overlaid onto [figure 8.6](#) as dashed lines with the colours corresponding to the respective values of D_m as shown in the figure legends. There is excellent agreement between the measured values and the expected theoretical value. Also, for $t \rightarrow 0$ we recover that the measured dispersion is just equal to D_m because the fluid motion has had no time to spread the tracers. This is visible on the lower plot, which has logarithmic axes.

Captured in [equation \(8.20\)](#) is the fact that at *lower* D_m we see a *higher* dispersion. The reason for this is that for higher diffusion coefficients the tracer particles sample the velocities within the pipe more quickly. This, in turn, means that the each tracer moves with a velocity closer to the average velocity in the pipe. Since there is less variation in the time averaged tracer velocities over a given time period they are spread out less (cf. [equation \(8.19\)](#)). At low D_m the tracers sit on pathlines with different velocities for a long time meaning that their time averaged velocities remain different for longer³. The pathlines for 200 tracer particles with different D_m are shown in [figure 8.7](#). These illustrate the fact that for higher D_m the particles move with a velocity which is closer to the average flow velocity in the channel thereby reducing the variance of the axial displacement.

8.4.2 Flow in a Pipe with Constriction

In the previous example the SPH particles formed laminae which flow past each other perfectly giving us $D_{\text{axial}} \sim t^1$ and $D_{\text{radial}} = 0$ exactly. As a more demanding test case we now consider the throttled pipe flow investigated in [section 5.3.1](#). In this case the flow geometry forces some lateral movement but because the flow is periodic we still expect that $D_{\text{axial}} \sim t$. Since there is some movement in the radial (lateral) direction due to the geometry initially D_{radial} will not be zero. This movement is bounded by the pipe, however, meaning that $\text{VAR}(u_{\text{radial}})$ is also bounded. Therefore as τ increases we expect this initial D_{radial} to decay to zero with $D_{\text{radial}} \sim \tau^{-1}$.

[Figure 8.8](#) shows the axial and radial dispersion coefficients as a function of dimensionless time interval. The curves show exactly the behaviour described above; for $\tau \lesssim 1$ we see an initially increasing D_{radial} which then begins to decrease as expected. The fitted curve gives $D_{\text{radial}} \sim \tau^{-0.9}$ which, while close to the expected theoretical value, does show some deviation (especially at larger time intervals). For the axial dispersion we see the fitted curve gives us $D_{\text{axial}} \sim \tau^{0.97}$ which is very close to the expected value of 1.

The slight departure from this scaling must be due to ‘numerical diffusion’. However, unlike numerical diffusion which affects other CFD methods this is not a spurious diffusion term arising from the discretization of advection terms. In this case the forcing of SPH particles

³At higher D_m the velocity auto correlation decays faster, see [section 8.5.4](#)

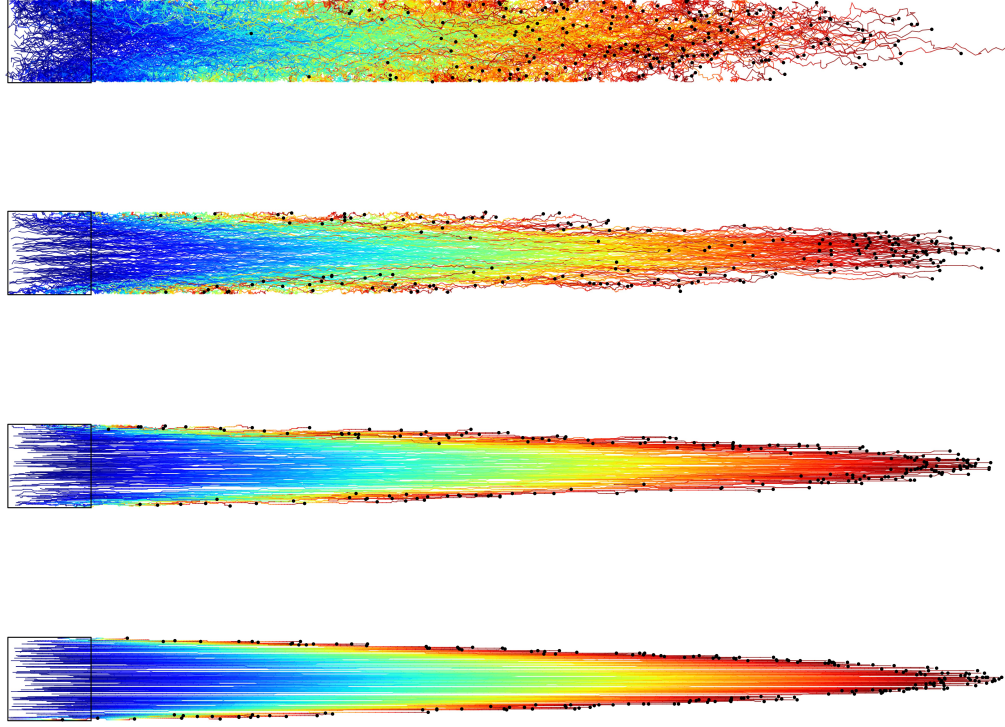


Figure 8.7: The pathlines for 200 tracer particles in plane Poiseuille flow after 6.25s for molecular diffusions of 10^{-5} , 10^{-6} , 10^{-7} and $10^{-8} \text{ m}^2 \text{ s}^{-1}$ from top to bottom respectively. The reduction in (axial) spread for higher D_m is clear. The colour represents the elapsed time with blue being $t = 0$ and red being $t = 6.25 \text{ s}$.

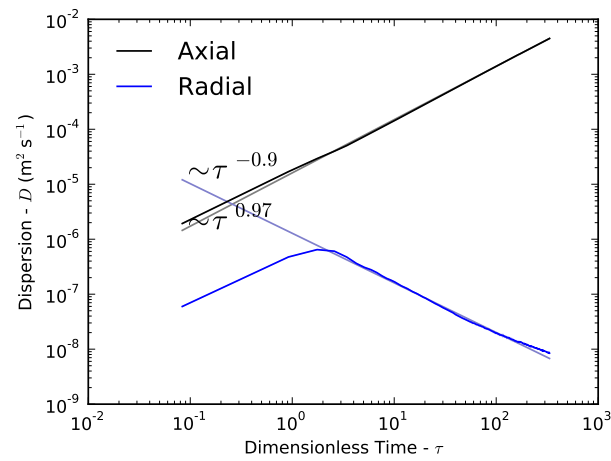


Figure 8.8: The axial and radial dispersion coefficients for flow in a periodic throttled pipe show close agreement to the theoretical trends $D_a \sim \tau$ and $D_r \sim \frac{1}{\tau}$.

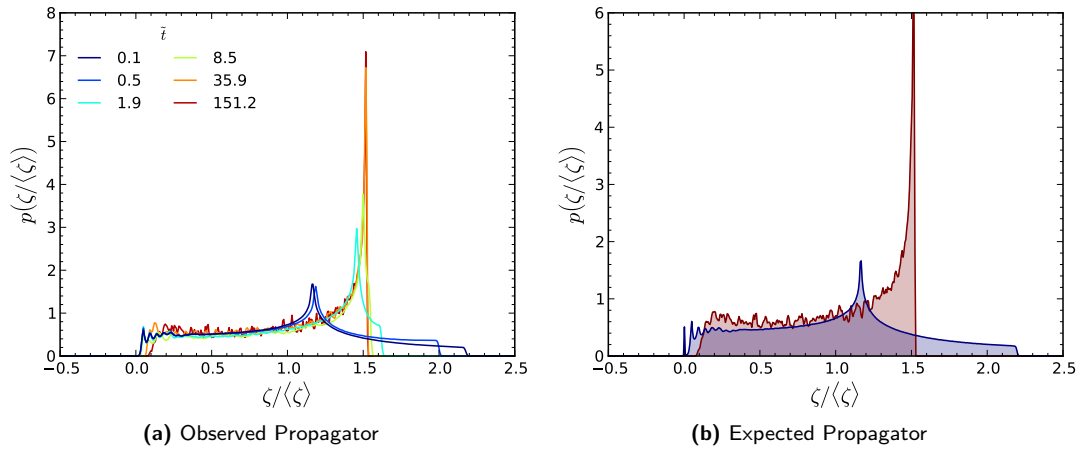


Figure 8.9: The propagator for flow in a pipe with constriction shown for various dimensionless times $\hat{t}$. At small times $p(\hat{\zeta})$ matches closely the distribution of velocities within the system which is shown as a blue fill. At later times the particles have had time to sample all the velocities on their pathline and so $p(\hat{\zeta})$ approaches the distribution of average particle velocities in the system which is shown as a red fill.

through the contraction causes particle disorder, then when the particles decelerate on exiting the constriction they might return to a slightly different pathline leading to a very weak breaking of our closed pathline assumption and hence a slight deviation from $D \sim \tau^1$. If this slight deviation from exactly Taylor behaviour is effectively random (i.e. it does not eventually cancel itself out somehow) we would expect to see a well defined dispersion coefficient (both axial and radial) for very long times. The reason we see a larger relative deviation for the radial dispersion is that, because it is smaller, any deviation in $\text{VAR}(u_{\text{radial}})$ will lead to larger relative effect than for axial dispersion.

The propagator $p(\hat{\zeta})$ for this example is shown for various times in [figure 8.9](#). For Taylor type dispersion the exact distribution (and corresponding variance) of the propagator depends on the distribution of velocities. If the velocities are distributed with a probability density function $f(v)$, with mean and variance μ and σ^2 respectively, then we find

$$p(\hat{\zeta}) = \mu f(\hat{\zeta}\mu), \quad (8.21)$$

which has no time dependence. If, as is the case here, flow is Taylor at long times but the velocities are not constant then we have

$$\lim_{t \rightarrow \infty} \text{VAR}(\hat{\zeta}) = \text{const.}$$

where the distribution tends to the distribution of average velocities in the system.

At short times the flow looks Taylor because the velocities of any give particle cannot have changed much which means for $\tilde{t} \ll 1$ we expect $p(\hat{\zeta})$ to be given by the (rescaled) distribution of all velocities throughout the system; highlighted in [figure 8.9b](#) as a blue fill. After some time $\tilde{t} > 1$ a particle will have seen all the velocities on its closed pathline and thus will have moved a distance given by $\langle v \rangle_{\text{path}} t$. This means that at later times the propagator approaches the rescaled distribution of *average* velocities in the system, which is highlighted as a red fill. This is also visible in [figure 8.8](#) as deviation of the measured curves from the fitted curves for times $\tilde{t} \lesssim 1$.

8.5 Regular Packing

As a first case we consider a regular hexagonal periodic packing in two dimensions whose geometry is shown in [figure 8.10](#). The solid particles have a diameter of 2 cm, which is representative of the sizes found in heaps. The water feels an acceleration due to gravity of 9.81 m s^{-2} . This is the only driving force for the systems and is equivalent to a pressure gradient of 9.81 kPa m^{-1} in a saturated system. For all unsaturated simulations the air phase feels no acceleration due to gravity because in the heap it would be in hydrostatic equilibrium (in the absence of air sparging). This can be achieved by applying an additional force in the direction opposite to $\mathbf{g}$ which exactly cancels the weight of the air, or, alternatively, by simply applying the gravitational force only to the water SPH particles. The first method is what happens in reality since a pressure gradient would be set up countering $\mathbf{g}$ and thus causing a slight buoyancy force on the

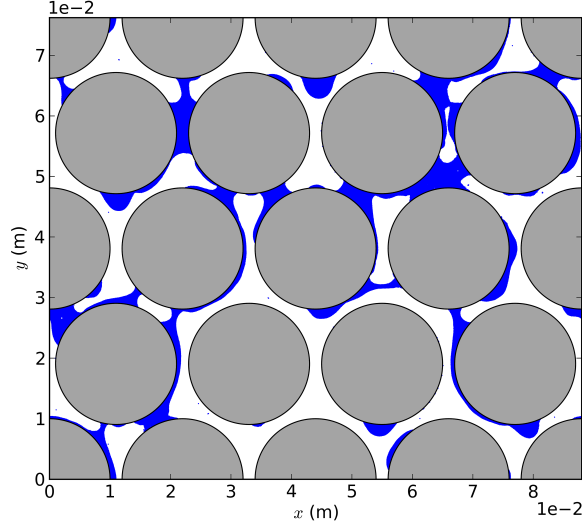


Figure 8.10: An example output of an unsaturated simulation for the regular periodic packing. Boundaries are shown in black, the aqueous phase in blue and the gaseous phase in white. This simulation has a water saturation of 22.5% and used 174×10^3 SPH particles.

water due to displaced air. As a fraction of the total gravitational acceleration this buoyancy force is

$$1 - \frac{\Delta\rho}{\rho_{\text{water}}}.$$

For the densities of an air-water system this contribution is 0.1%, i.e. so small that applying the second method is valid approximation. Therefore this is the method used here.

The equilibrium contact angle used for all simulations in this chapter is $\theta = 30^\circ$ which is representative of the contact angle seen in real heaps [53].

8.5.1 Flow Structures

Depending on the saturation, θ_s , one can observe very different flow structures in our unsaturated simulations. Example outputs from simulations at a range of saturations are shown in figure 8.11.

At the lowest saturations, $\theta_s \lesssim 10\%$, the fluid exists as discrete droplets which are so small that the force of gravity is not able to overcome the capillarity and the fluid is held up. For slightly larger saturations the fluid still exists primarily as droplets but there is sufficient fluid that these can flow through the system, combining with other droplets as they proceed through the domain as rivulets. These rivulets can break down as some of their fluid becomes trapped at capillary junctions. At these low saturations there is little coupling between the fluid and the gas phase with the gas phase having no net downward motion.

Eventually when the saturation reaches $\approx 50\%$ the gas becomes highly entrained leading to strong coupling between the phases. At this point the fluid in the domain starts to move

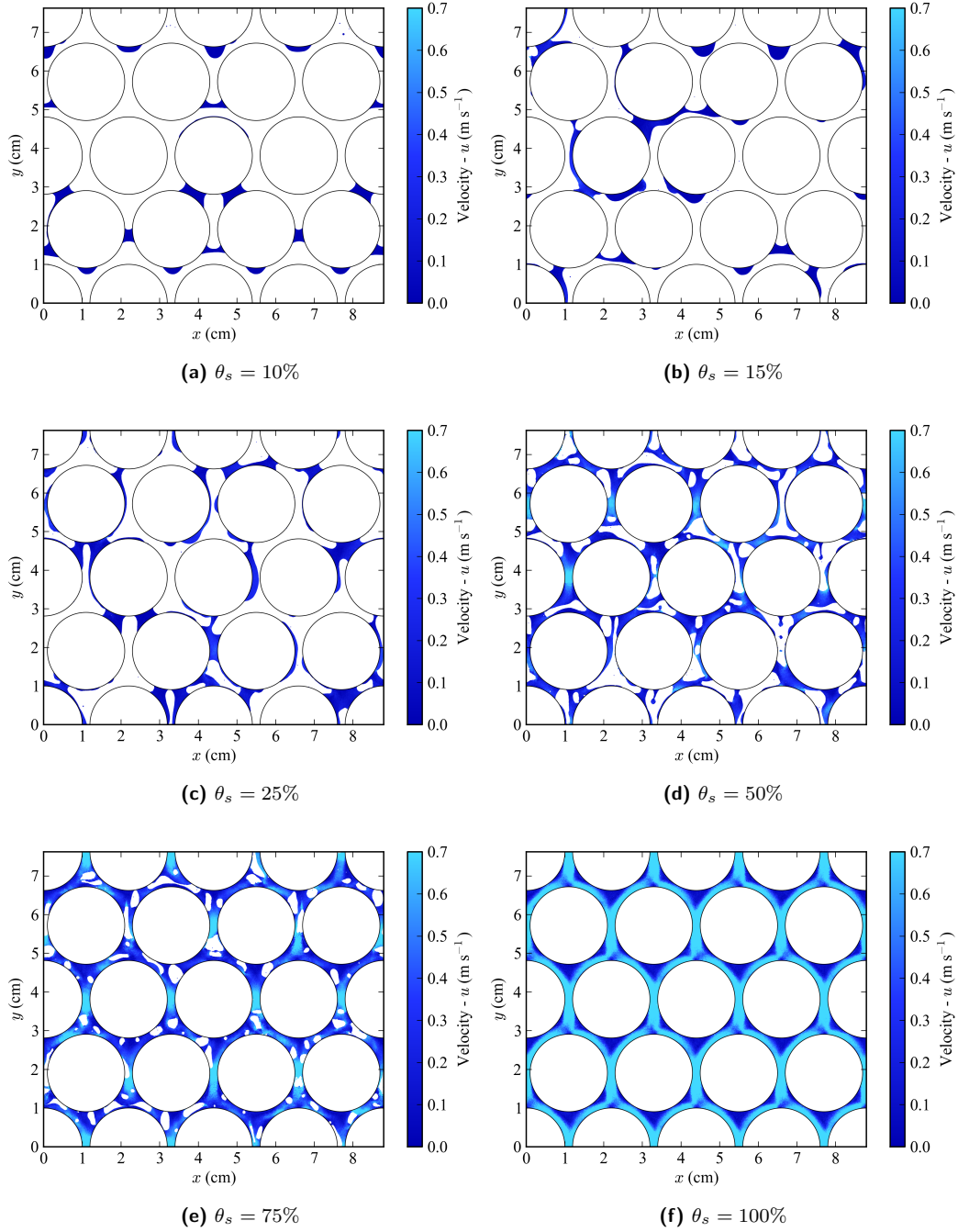


Figure 8.11: Depending on the saturation of the system one encounters very different flow structures. At very low saturations the flow is discrete droplets and is completely held up by capillarity. As saturation increases these begin to flow and mix with one another until eventually the fluid takes up more of the voidage and one can see entrained gas bubbles.

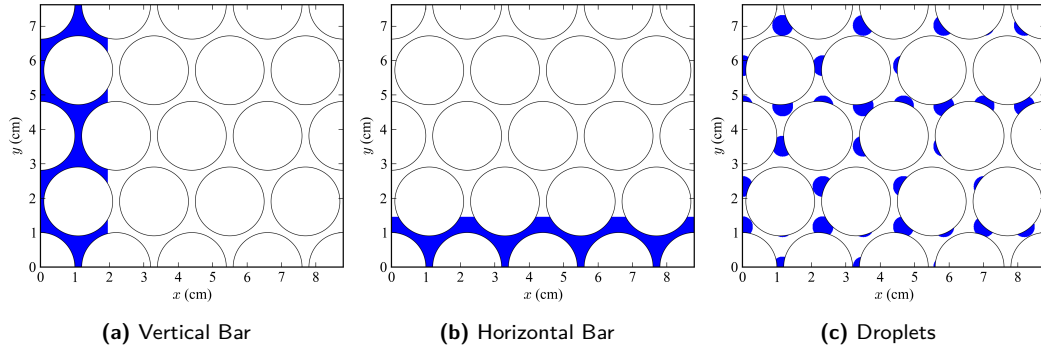


Figure 8.12: The different initial fluid configurations tested. The examples shown here have saturation $\theta_s = 20\%$.

generally together as the coupling through the gas phase acts to increase the correlation of flow structures in the system. At the highest saturations that are not fully saturated there are only a few gas bubbles, which get pulled along with the fluid. They still play an important role, however, by disrupting the flow and leading to the unsteadiness in the simulations. Once the system is fully saturated inertial effects mean that the flow is not entirely steady, but without the entrained gas bubbles this unsteadiness is much reduced compared to the almost saturated cases.

8.5.2 Hysteresis and Initial Conditions

It is known that packed beds exhibit strong hysteresis and sensitivity to initial conditions (wet vs. dry) [52, 53] therefore the initial conditions of the simulations are expected to affect the course of the simulation. To test the role this could play in the simulations, the water was placed according to three different initial arrangements (for a range of water saturations). The initial water configurations are

- A vertical strip spanning the domain period in the gravity direction.
- A horizontal strip spanning the domain perpendicular to the gravity direction.
- A series of droplets of radius >3 mm distributed through the domain.

These configurations are shown in figure 8.12 for a saturation of $\theta_s = 10\%$. A radius of >3 mm was chosen to make the droplets somewhat larger than the capillary length⁴ $\lambda_c = 2.7\text{mm}$ without making them larger than necessary. As the saturation approaches 100% the differences between the three configurations will become less until finally at 100% they become degenerate.

We can see in figure 8.13 that for 10% saturation, the behaviour of $v_s(t)$ is markedly different for the three configurations. In the horizontal strip case the fluid is held at the junctions by capillarity and there is not enough weight of fluid above these to force the fluid through. Meaning initially the system is in equilibrium and nothing happens, this is very different behaviour to the vertical strip and droplet cases where the fluid initially accelerates then reaches a roughly

⁴ $\lambda_c = \sqrt{\frac{\alpha}{\rho g}}$

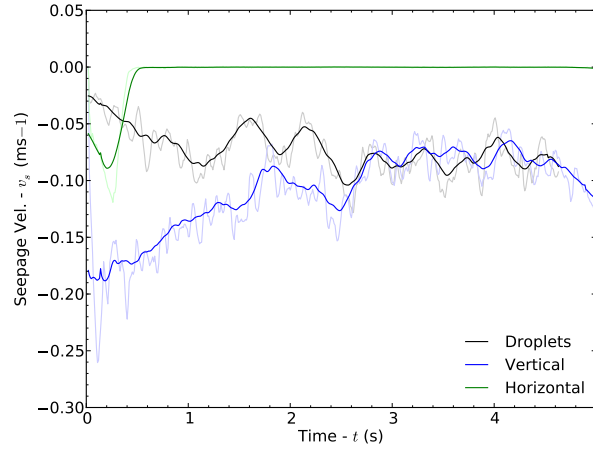


Figure 8.13: The seepage velocities for the three different ICs at a saturation of 10% show that the initial condition can have large effects on the long term behaviour of the system.

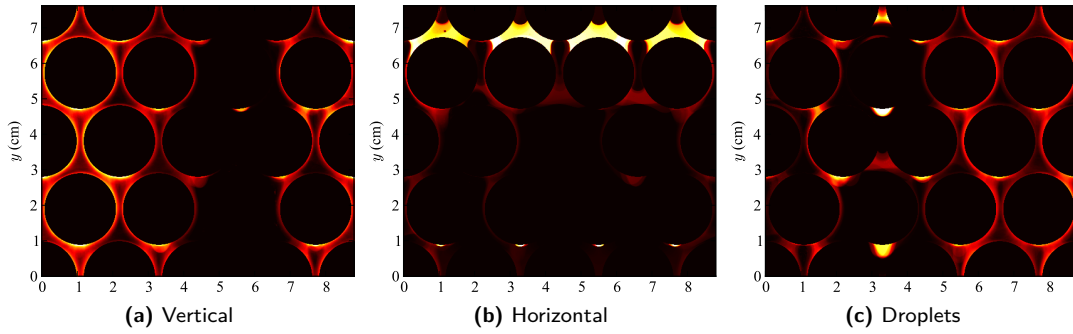


Figure 8.14: The flow forms channels & rivulets at low saturations as can be seen from the occupancy plots. The horizontal case is initially held at capillary junctions of the particles as can be seen in this plot.

steady seepage velocity of -0.1 ms^{-1} . Further, the flow in these two cases remains localized. While this may not be surprising for the initially vertical configuration, the droplet case did not start out with more fluid in one region than another. Therefore this development is due to the forces at play in the system. This localization can be seen in [figure 8.14](#) which shows the ‘occupancy’ of the each point in the domain. It is effectively a 2D histogram representing how many times an SPH particle was found at a particular location throughout the course of the simulation.

This localization of flow into a preferred channel proceeds differently for the two flowing cases, as can be seen from the different profiles for $v_s(t)$ ([figure 8.13](#)). Initially the vertical strip accelerated quickly to a fast flowing regime because the local saturation is high at this point, but as the simulation progresses fluid moves away from this region and is held there, in part, by capillarity. As the fluid spreads the local saturation decreases along with the flow velocity until it reaches -0.1 ms^{-1} . For the droplet case the saturation is initially $\approx 10\%$ throughout the domain. In this case the fluid accelerates up to -0.1 ms^{-1} without over shooting as the

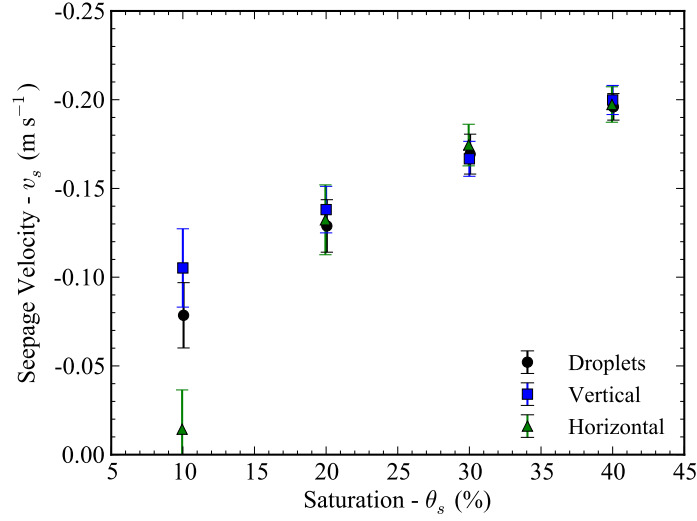


Figure 8.15: The long time seepage velocity for the different initial conditions at various saturations shows that for $\theta \gtrsim 10$ the initial condition has little effect on the resulting flow velocity.

capillary forces cause the flow channel to establish itself.

figure 8.15 shows the seepage velocity for the different initial conditions at various saturations. From this it is apparent that for $\theta \gtrsim 20\%$ the initial condition has no discernible effect on the resulting v_s . However we have seen from figure 8.14 that the initial dynamics can be quite different. Because the droplet initial condition is closer to the long time fluid distribution, unless otherwise stated, further simulations in this thesis will use the droplet initial configuration.

8.5.3 Dependence on Saturation

Since our simulations have neither in- nor outflow boundaries we *cannot* set the flow rate. Instead because of the periodicity in our simulations we set the amount of liquid in the system, which stays constant, and measure the resulting flow patterns. In effect performing the reverse of [53] wherein the flow rate is specified and resulting holdup and dispersions are measured. Therefore the dependence on saturation of our systems is a key variable of interest.

Seepage Velocity

In section 8.5.2 we saw that for a saturation of $\theta = 10\%$ with the droplet configuration there was a non-zero seepage velocity. However in the simulations shown in this section a slightly different spacing was used for the droplets. In this case the slight difference leads to a situation where the droplets all become held at capillary junctions and no flow channel establishes itself. This serves to emphasize just how sensitive these systems are to their history. The seepage velocity as a function of saturation is shown in figure 8.16. It shows that the seepage velocity

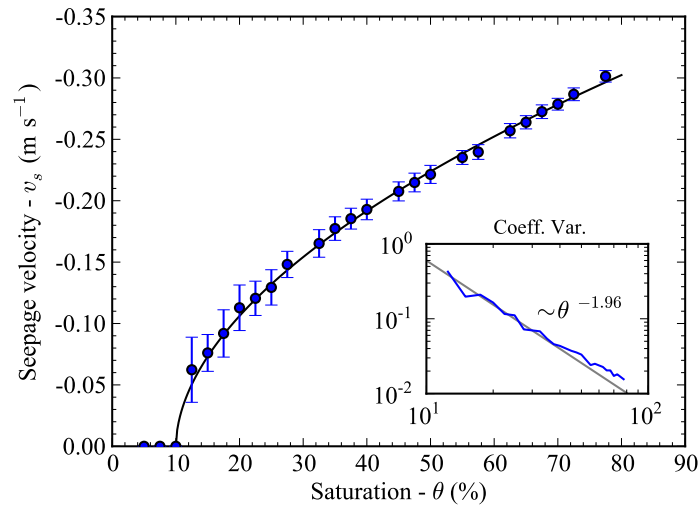


Figure 8.16: The seepage velocity v_s for a regular packing of circles of radius 1 cm for different saturations θ . Below a certain saturation the fluid is entirely held up and the seepage velocity is zero. Inset shows the coefficient of variation of the velocity with time, at the lowest saturations where we have discrete droplets we see largest CV.

does not steadily increase from zero at some critical saturation. The reason for this is that once above a certain saturation there is sufficient fluid to form flowing channels in which the flow velocity is comparatively high. However the average seepage velocity is reduced by the fact that some of the fluid becomes held up.

Additionally at these low saturations the flow in the flowing channels proceeds in the form of droplets. As a droplet flows through the system it must flow around obstacles, through capillary junctions, etc; this causes the average seepage velocity to vary a lot with time. This can be seen by looking at the coefficient of variation of the seepage velocity with time

$$CV = \frac{\sigma}{\mu}$$

where σ and μ are the standard deviation and mean, respectively, of v_s over time. The coefficient of variation is shown in [figure 8.16](#) (inset). It follows the trend $CV \sim \theta^{-1.96}$ according to a best fit. Since the variability in the seepage velocity seems to be linked to the transit of droplets through the system one might expect that there is a value for θ_s beyond which CV suddenly drops. Instead at moderate saturations the fluid flows as ‘plugs’ which are separated by bubbles of entrained gas. This gas has the effect of causing variability in the velocity as it passes through the capillary junctions. As θ_s increases, and the gas occupies less of the voidage, more junctions are saturated meaning the fluid is not so perturbed as it progresses through the periodic media. This is, however, primarily a feature of the 2D simulation domain because the air phase cannot pass around rivulets and droplets as they move in the narrow capillary junctions. Instead they are forced along with the flow. This, therefore, would likely be quite different in a 3D case.

Dispersion at Low Saturations

The difference in flow behaviour at lower saturations also manifests itself in different behaviours for the two dispersion coefficients (axial and radial) at different saturations. The real dispersion coefficients against time interval are shown in [figures 8.17a](#) and [8.17b](#) and their rescaled (dimensionless) equivalents are shown in [figures 8.17c](#) and [8.17d](#). Despite the real coefficients being orders of magnitude different we can see that rescaling to the dimensionless quantities allows us to compare nicely the behaviour between systems with different saturations. The final values for the measured dispersion coefficients are shown in [figure 8.18](#).

For both axial and radial dispersions, at very small τ we can see the dispersion following $D \sim \tau$. This is an artefact of the method by which D is calculated. For very small τ a particle’s velocity cannot have changed by a lot; the correlation between $\mathbf{v}_i(t)$ and $\mathbf{v}_i(t + \tau)$ is high. Since each particle has a different velocity, over short times we see Taylor dispersion. As $\tilde{\tau}$ approaches 1 dimensionless time unit this correlation has reduced significantly meaning a particle could potentially be moving in any direction. This explains why for $\tilde{\tau} > 1$ we observe a change in the slope of $-\log(\text{Pe}_d)$ versus $\log(\tau)$.

Focusing on the axial dispersion it clear that the trends have a strong dependence on θ . For the lowest saturation D reaches an asymptotic value after ~ 20 dimensionless time units and this

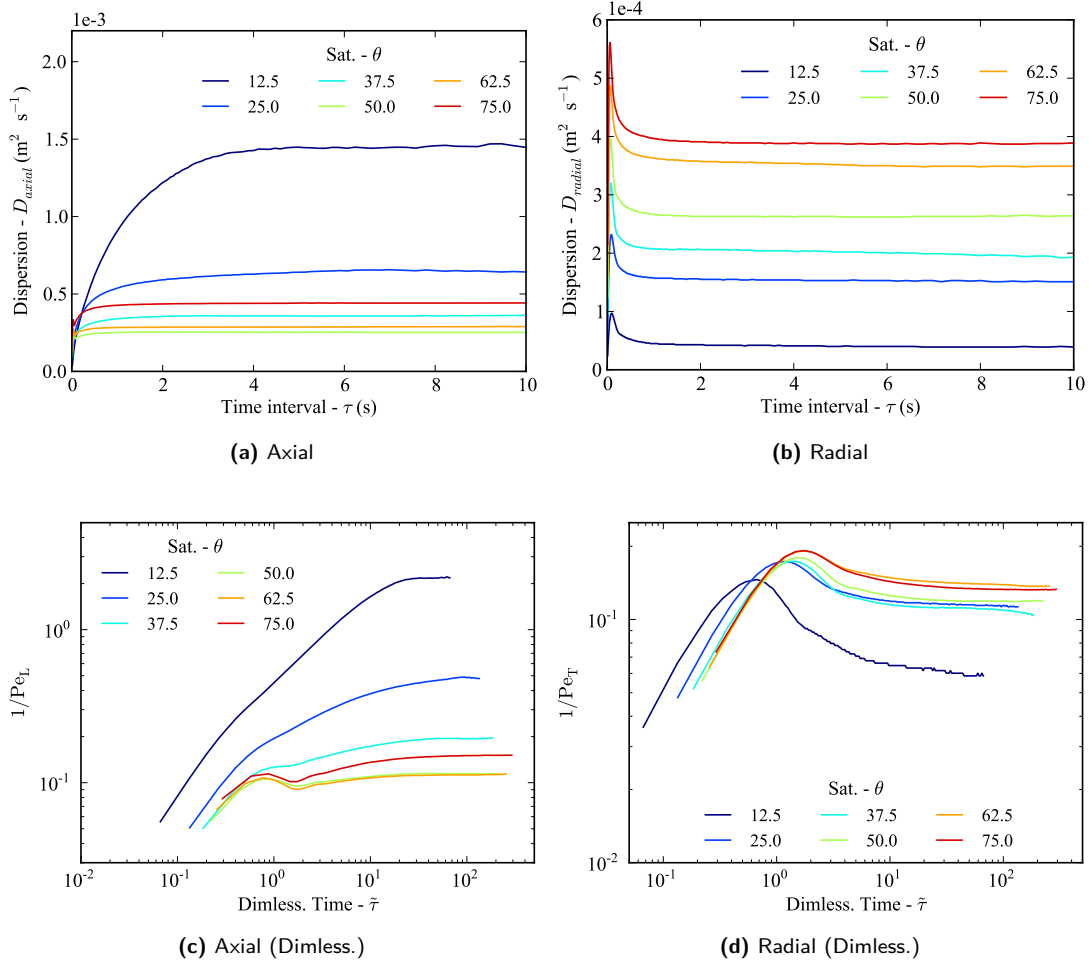


Figure 8.17: The radial and axial dispersion coefficients in a regular packing plotted for various saturations against time and the rescaled (dimensionless) equivalents.

value is much higher than for higher saturations (except completely saturated). This increased dispersion is mainly caused by static droplets. To see why, consider the flow being in two distinct populations; flowing and static. One can show quite easily that the static droplets enhance axial dispersion. Let the static fluid make a proportion $p \in [0, 1]$ of the total fluid and image that the moving fluid causes a dispersion $D = C\tau^a$ where $a \geq 0$. $a = 1$ corresponds to Taylor dispersion (section 8.4). It can be shown⁵ that the dispersion for the overall population is given by

$$D(\tau) = C(1-p)\tau^a + (v'_s)^2 p(1-p)\tau \quad (8.22)$$

where v'_s is the seepage velocity of the flowing component *only*. The overall seepage velocity is $v_s = (1-p)v'_s$ which is less than v'_s . Equation (8.22) does not capture the effect of high velocity autocorrelation at small times since it assumes the dispersion in the moving phase $\sim \tau^a$ even at small times. If there is no mixing between these two populations and the dispersion in the flowing channel is not faster than Taylor ($a \leq 1$) then we recover Taylor dispersion at long

⁵See appendix A.7

times, $D \sim \tau$, because eventually the second term will always dominate the first. When exactly the term begins to dominate depends on the magnitude of the dispersion in the flowing phase C and the proportion of the fluid which is flowing $(1 - p)$. If p is zero then we recover the dispersion caused purely by the flowing channels and if p is one then we get, as expected, zero dispersion because nothing is moving.

Figure 8.19 shows the pathlines of the fluid for $\theta = 12.5\%$ and $\theta = 25.0\%$ where the colour represents the elapsed time. It is clear that the behaviour is qualitatively different for these two cases. In figure 8.19a we see how the presence of droplets affect the pathlines; the times at which different parts of the flow reach a certain depth are very different as compared to figure 8.19b where the time each part of the flow sees a particular depth is (roughly) constant, as indicated by the approximately constant colour gradient of the pathlines from left to right. In our simulations there is some mixing between the flowing fluid and the held up droplets as can be seen in the pathlines. This mixing is the reason that we observe $D \sim \tau^{0.4}$ (instead of Taylor dispersion) for $1 \lesssim \tilde{t} \lesssim 10$, and eventually $D = \text{const.}$ for $\tilde{t} \gg 10$. Despite this mixing though, it is the (in)action of the droplets which enhances the dispersion in our periodic medium for lower saturations.

Propagators

This difference in behaviour can also be seen in the propagators, which are shown in figure 8.20. For the higher saturations (25% shown) we find that at later times the propagator $p(\hat{\zeta})$ is well approximated by a Gaussian distribution with decreasing variance, which indicates Fickian dispersion. For the lowest saturation, $p(\hat{\zeta})$ is still very far from being Gaussian when, after the same time, $p(\hat{\zeta})$ for higher saturations is already well approximated by a Gaussian. The peaks in the propagator correspond to displacements in multiples of the characteristic length L . At low saturations they occur because the flow consists of discrete droplets. The droplets flow quickly around the particles in the system but are then held at the bottom of the particles before the droplet becomes large enough to bridge to another particle or detach due to its weight. Thus the most likely displacement is $\sim L$. The fact there are more peaks which are closer together as time increases is due to the rescaling; $\langle \zeta \rangle \approx v_s t$ meaning the peaks in $p(\hat{\zeta})$ are at $\frac{nL}{v_s t}$. Interestingly, even at the highest saturations where the flow is effectively contiguous with bubbles of the gas phase we see these peaks. In this case it is because between the particles the fluid flows much quicker, and therefore particles spend less time in the regions between the particle layers, and thus the most likely displacements are still at multiples of L . The first effect is an artefact of the discrete nature of the flow whereas the second is an artefact of the regular geometry of the system. This suggests that for a mono-disperse random packing we might still expect to see the peaks at low saturations but that at higher saturations the disorder in the packing might act to smooth the peaks somewhat.

The radial dispersion (figure 8.17d) shows similar behaviour for all saturations except the very lowest. Not only is the dispersion lower by almost an order of magnitude its dependence on $\tilde{\tau}$ is also different. The higher saturations approach an approximately constant $\tilde{D}$ after $\tilde{\tau} \approx 1$ whereas for $\theta = 12.5\%$ we observe that D is decreasing slightly even as $\tilde{t}$ approaches 10^2 .

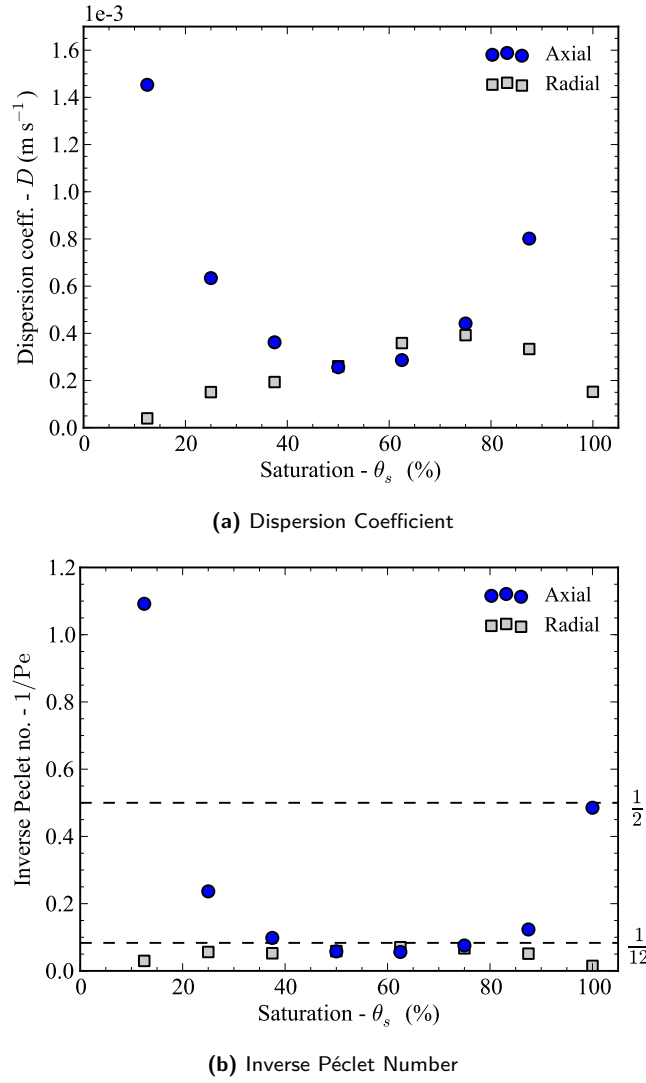
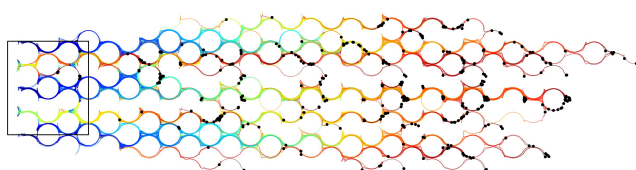
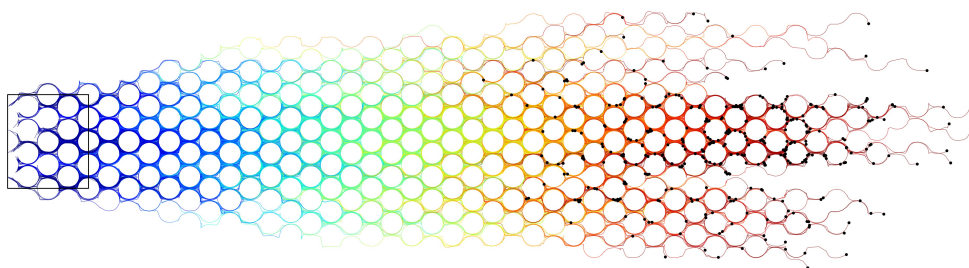


Figure 8.18: The final measured dispersion values and inverse Péclet numbers as a function of the system's saturation θ_s . The Theoretical asymptotic values for saturated systems as $\lim 1/\text{Pe}_m \rightarrow \infty$ for the axial and radial inverse Péclet numbers are given in [32] as $1/2$ and $1/12$ respectively, these are highlighted on the graph with dashed lines.



(a) $\theta = 12.5\%$



(b) $\theta = 25.0\%$

Figure 8.19: The pathlines of 400 SPH particles show qualitatively different behaviour at saturations of 12.5% and 25.0%. The colour represents the elapsed time with blue being $t = 0$ s and red being $t = 5$ s.

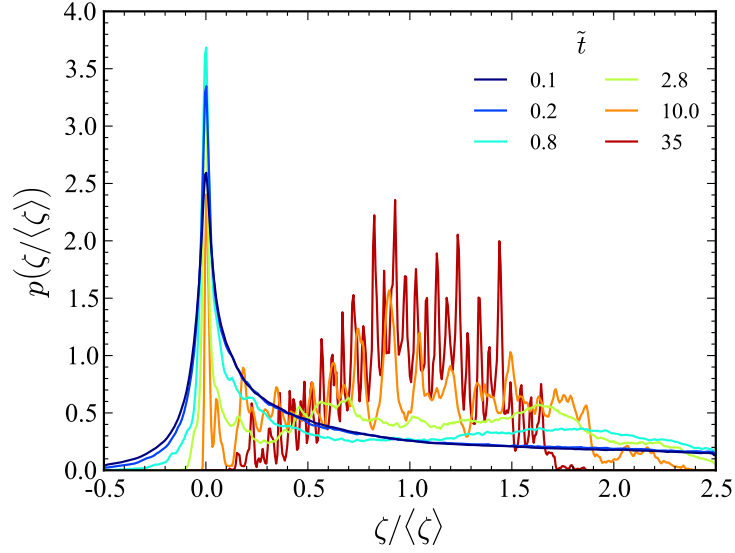
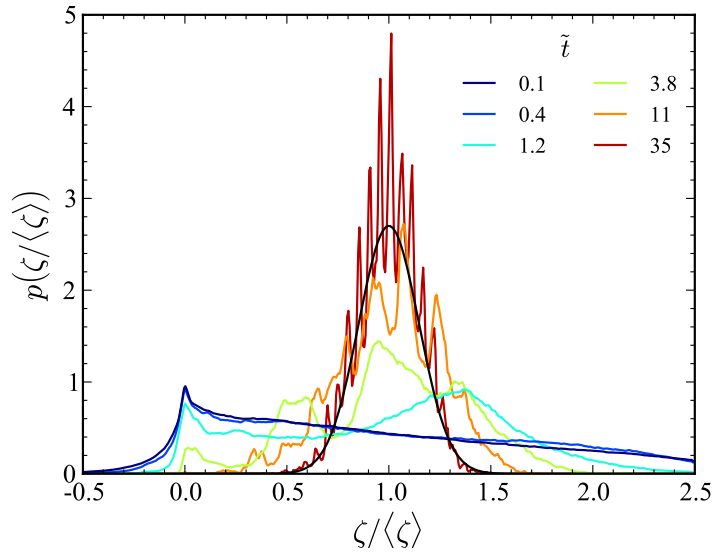
(a) $\theta = 12.5\%$ (b) $\theta = 25.0\%$

Figure 8.20: The propagator $p(\hat{\zeta})$ shown at different times for two saturations; 12.5% and 25.0%. We can see that for the higher saturation at later times the propagator can be seen to be approximately Gaussian indicating the approach to Fickian dispersion but at the lower saturation even at the longest time reached by the simulation $p(\hat{\zeta})$ is poorly approximated by a Gaussian distribution. The peaks in the distribution arise because fluid is more likely to have moved one particle displacement than any other distance.

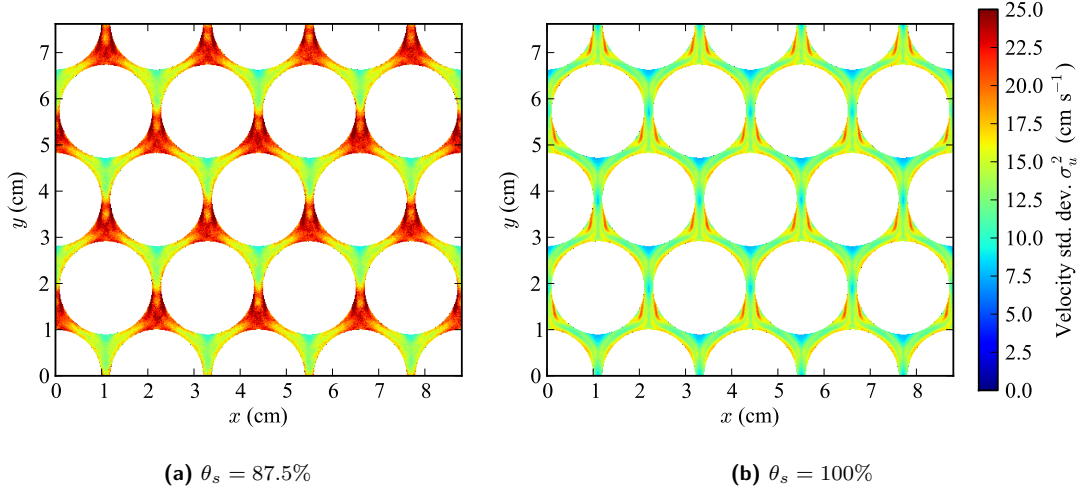


Figure 8.21: The standard deviation of the velocity over the simulation domain shows that for the completely saturated case the velocity, while not completely steady, has much less variation than for the case with entrained air bubbles. This variation leads to a large reduction in the axial dispersion.

Equation (8.22) can be applied to radial dispersion by noticing that (necessarily) $v'_s = 0$ meaning we get $D(\tau) = C(1 - p)\tau^a$. In this case the static droplets act to reduce the dispersion by a factor $(1 - p)$ but they do not change the dependence of D on τ . It seems quite probable that this is also the mechanism causing a different D for $\theta_s = 12.5\%$.

Dispersion at High Saturations

In Delgado [32] the author gives an empirical relationship between the molecular and dispersion Péclet numbers in completely saturated packed beds as

$$\frac{1}{\text{Pe}_L} = \frac{1}{T} \frac{1}{\text{Pe}_m} + \frac{1}{2}, \quad \frac{1}{\text{Pe}_T} = \frac{1}{T} \frac{1}{\text{Pe}_m} + \frac{1}{12}$$

where T is the flow path tortuosity. Since our simulations, in the base case, do not consider molecular diffusion, we are simulating the behaviour in the limit $1/\text{Pe}_m \rightarrow 0$. Thus for a saturation of $\theta_s = 100\%$ we expect to recover $1/\text{Pe}_L = 1/2$ and $1/\text{Pe}_T = 1/12$. We can see from figure 8.18 that at the highest saturation we recover a value very close to $1/2$ for the axial dispersion as expected. The radial Péclet number however is quite some way from the value of 12 which would be predicted by the empirical relationship.

At the next highest saturation ($\theta_s = 87.5\%$) the axial dispersion is far lower. The dimensionless dispersion is $0.12 \ll 0.5$. This dramatic reduction in dispersion is caused by the gas bubbles which become entrained in the flow. As the bubbles pass through a capillary junction they do not split into smaller bubbles, instead surface tension acts to hold them together. Thus by the bubble passing down one possible path rather than the other this introduces a large degree of randomness which is not found in the completely saturated case. This is reflected

in the standard deviation of the velocity at the capillary junctions, plotting this for the two systems we can see that the velocity, while not completely steady in the 100% saturated case, has a far lower standard deviation at the capillary junctions than in the almost saturated case. This large variability of the velocity has a similar effect to the large molecular diffusions in [section 8.4](#); it mixes the fluid velocities and means that any fluid that is in a slow moving part of the domain is regularly mixed with fast moving fluid thereby ensuring that for each fluid particle its velocity, over time, is much closer to the seepage velocity. Since the variability in the velocity at capillary junctions is lower in the saturated case we see a far higher value of D .

8.5.4 Velocity Autocorrelation

The velocity auto-correlation quantifies how likely we are to see the same velocity for an SPH particle parcel after a given time interval has elapsed. Mathematically it is defined as

$$\phi(t) = \frac{1}{\sigma_v^2} \left\langle \int_{-\infty}^{\infty} (v(\tau) - v_s) \cdot (v(\tau + t) - v_s) d\tau \right\rangle, \quad (8.23)$$

where $\sigma_v^2 = \text{VAR}(v(t))$ and the average is a volume average over the fluid in the domain. Its utility lies in that it can be related to the hydrodynamic dispersion by the Green-Kubo relation:

$$D = \sigma_v^2 \int_0^{\infty} \phi(\tau) d\tau, \quad (8.24)$$

the derivation and implications of which are discussed in [\[62\]](#). The derivation is reproduced in [appendix A.8](#).

In [equation \(8.23\)](#) the velocities first have the seepage velocity v_s subtracted, if this were not done, then even at long times ($t \rightarrow \infty$) we would observe a non-zero velocity correlation and the integral [equation \(8.24\)](#) would diverge. Since the fluid in our system is inertial, at short times ($\tilde{t} \ll 1$) we would expect a high auto correlation which drops off after this.

[Figure 8.22](#) shows the velocity autocorrelation function for the axial velocity at a range of saturations. The large variability at long times is again caused by the lack of samples due to the finite run time of the simulations. The difference in behaviour for the different simulations is clear; larger saturations decorrelate with a time of $\tilde{t} \approx 1$ which corresponds to having sampled the velocities on a path around a particle, this echos the kink seen in the plots of [figure 8.17c](#) at $\tilde{t} = 1$. For the highest saturations we even observe that ϕ becomes negative at ≈ 1 . This implies that if we are moving with a velocity $> v_s$ then at a time $\tilde{t} = 1$ later we are likely to be moving with a velocity $< v_s$ and vice versa if we are moving with a velocity $< v_s$. This is due to the fact that fluid flows faster in the vertical channels through the packing than the off vertical channels.

Of particular interest is $\phi(t)$ for the lowest saturation, $\theta = 12.5\%$. Here the velocity remains correlated even beyond $\tilde{t} = 1$. This is largely due the stationary droplets. If a fluid parcel remains motionless (or has any other constant velocity) then its velocity autocorrelation is necessarily independent of time and equal to unity. The presence of stationary droplets which

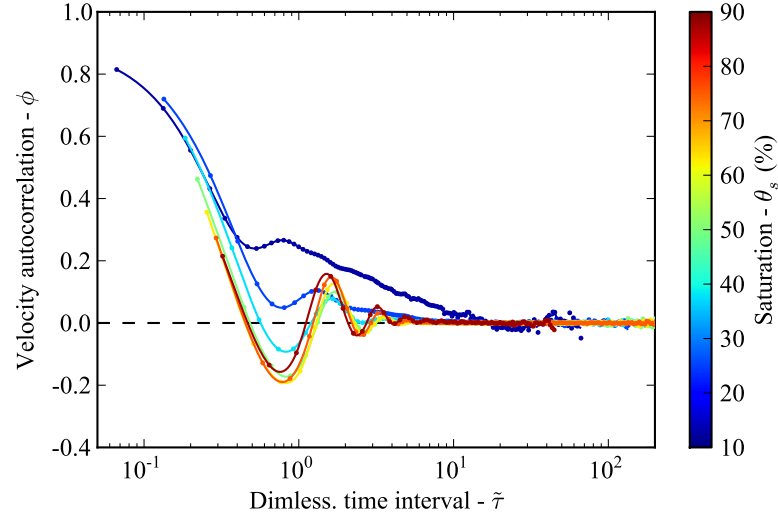


Figure 8.22: The velocity auto-correlation function is shown for several saturations. At short times ϕ is large due to the inertial nature of the system. As the dimensionless time approaches unity, the velocities begin to become uncorrelated for the higher saturations, even becoming negative. But for the lowest saturation ϕ remains high for times beyond $\tilde{t} = 1$, largely because of the stationary droplets, which if they remained indefinitely would have $\phi(t) = 1$.

Table 8.1: The gap width and corresponding Bond number.

d_c (mm)	Bo (1 dp)
1	0.2
2	0.8
3	1.9
4	3.3
5	5.2
6	7.4

persist in the system for extended time has the effect of increasing the auto-correlation at longer times due to this fact. As explained in [section 8.5.3](#), the reason that droplets act to enhance the axial dispersion is due to the fact they are stationary (see [equation \(8.22\)](#)). Considering the autocorrelation encodes the same information but provides a different viewpoint. The droplets increase ϕ at longer times (if there is no mixing, indefinitely), which in turn leads to an increase in $D(t)$ through the integral in the Green-Kubo relation, [equation \(8.24\)](#). Simply stated: if things are moving at the same velocity, then they are not spreading apart.

8.5.5 Flow Channel Width

In the previous section the distance between the packing particles at the nearest approach was 2 mm, which is comparable to the capillary length of water at 50°C, which is 2.25 mm. In this section we investigate the behaviour of the same system but with the gap between particles, d_c , varying from 1 mm to 6 mm at a saturation of $\theta_s = 15\%$. The Bond number for these systems varies widely since $\text{Bo} = (d_c/\lambda_c)^2$ varies with the square of d_c , where we are taking

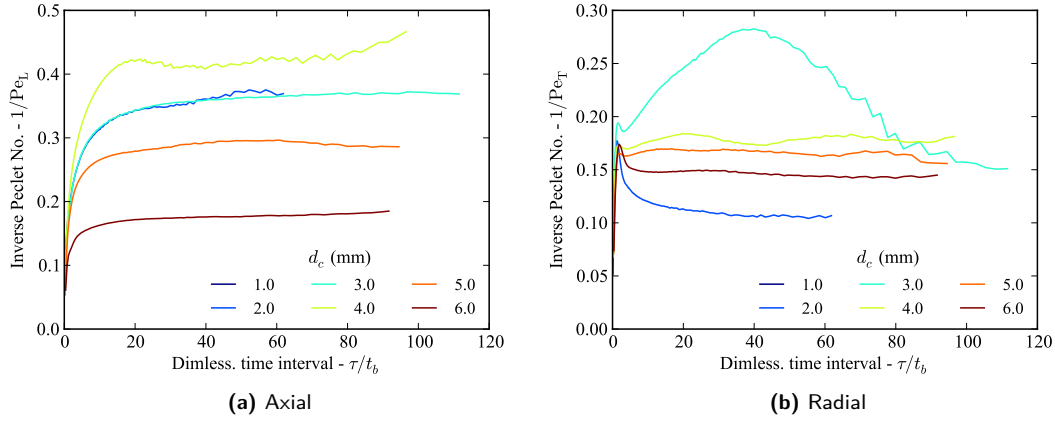


Figure 8.23: The axial and radial dispersion coefficients as a function of $\tilde{\tau}$ for different d_c . For the lowest d_c the system is capillary dominated and we get a very different behaviour from larger d_c s.

the channel width as the characteristic length scale. The Bond number for each gap width is shown in [table 8.1](#). For $d_c = 1$ mm the Bond number indicates that the dominant force should be capillarity. This is indeed the case, as the fluid does not flow and is entirely held at the capillary junctions. For $d_c > 1$ mm flow does establish itself. The resulting dimensionless dispersion coefficients (inverse Péclet numbers) are shown in [figure 8.23](#). There does not appear to be a clear trend in the values adopted by the axial dispersion for long times.

Examining the velocity of the flow for the different d_c 's reveals an interesting interplay between the geometry of the system and the forces at play. The capillary forces are completely symmetric, i.e. they would be the same under any arbitrary affine transformation of our system. Gravity obviously breaks this symmetry by imposing a preferred direction of flow on the system, but even so, because our systems are regular and periodic, the system should still display reflection symmetry in any line parallel to gravity.

For both the lowest and highest flowing Bond numbers (0.8 and 7.4) the flow is symmetric under the reflection operation. However, for the other intermediate values we see a symmetry breaking. It is most pronounced for $d_c = 4$ mm, whose velocity profile is shown in [figure 8.24](#). Each point in the domain is coloured according to the average velocity magnitude of the fluid phase seen there. The flow is, on average, not flowing in the direction of gravity (the negative y direction). This flow pattern establishes itself due to the persistence of flow channels once they are established. There is enough fluid in the system that it can form rivulet flow but because the system is not entirely capillary dominated inertia plays a role. As a droplet flows down one side of a circle it is carried past the bottom and the inertia carries it onto the droplet below and to the right, after which the droplet flows down the right side of this circle, and so forth. Interestingly this symmetry breaking does not occur immediately. For $d_c = 3$ mm the flow is initially parallel to the gravity direction, but after approximately 3 s the symmetry breaking occurs and the flow starts moving along the packings line of symmetry. This manifests itself as a large deviation in the measured value for D_{radial} , and therefore $1/\text{Pe}_T$, which can be seen in

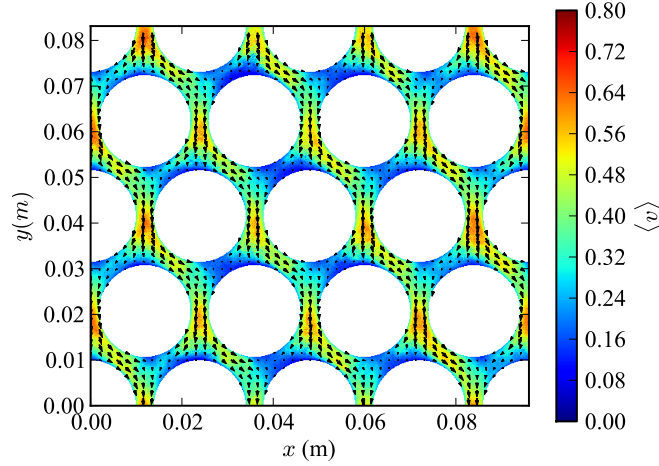


Figure 8.24: The average velocity of the water phase over time, throughout the simulation domain for a particle gap of $d_c = 4\text{mm}$ shows a breaking of reflection symmetry. The geometry of the system means that the average flow is not parallel to gravity (in the $-y$ direction).

figure 8.23b.

8.6 Effects of Geometry

The exact geometry of the packed beds can have a large effect on the flow characteristics. In this section we examine the effect of changing the orientation of the regular packing and compare results to irregular packings of monodisperse circles to check the sensitivity of the results to the exact geometry used.

8.6.1 Particle Radius

The radius of the particles used to generate the packing will have a large effect on the dynamics of the system. Clearly a droplet can only be held up if it is between two particles which are less than the capillary length scale apart. A very rough calculation gives the fraction of the particle surfaces which are $< \lambda$ away from another particle as

$$\frac{1}{2\pi} \arcsin \left(\frac{1}{2Rl} \sqrt{4l^2R^2 - (l^2 - \lambda^2 + R^2)^2} \right) \propto \frac{1}{R}, \quad (8.25)$$

where $l = R + p$ is the radius of the particles R plus the particle padding p . This suggests that as the particle radius decreases we will see a lower superficial velocity due to the hold up of fluid at capillary junctions. The effect of this on the dispersion is harder to predict. Figure 8.25 shows the measured seepage velocity for various particle diameters - D_p . As expected v_s increases with increasing D_p . An interesting question to ask then is, how does the seepage velocity affect the dispersion? According to the usual equations used in continuum scale models the dispersion coefficient is proportional to the flow velocity (c.f. equation (2.6)). Figure 8.26 shows the

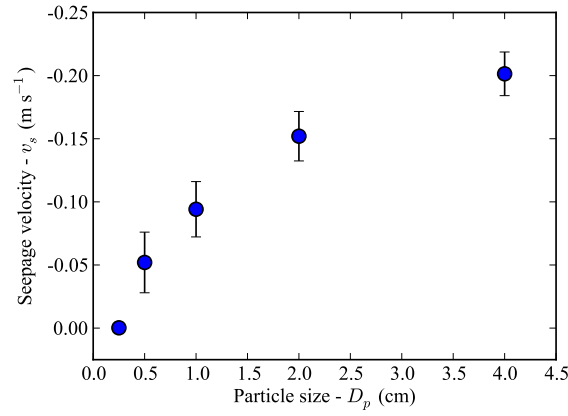


Figure 8.25: The seepage velocity is plotted against the particle diameter used for a regular monodisperse packing. As expected a lower D_p leads to a lower seepage velocity as more of the fluid is held between the particles.

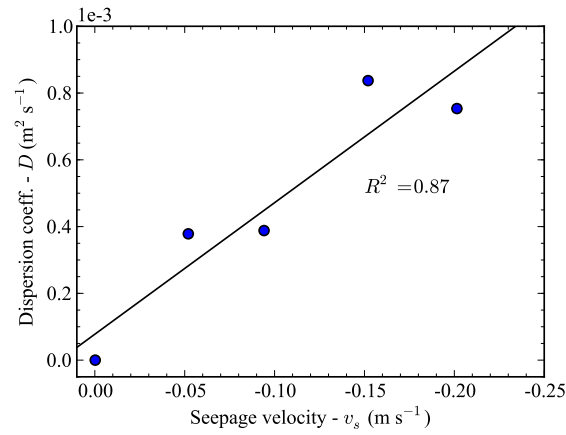


Figure 8.26: The measured dispersion coefficient is plotted against the measured seepage velocity which arises in systems with differing particle radii.

measured dispersion coefficient plotted against v_s . A linear regression has been used to fit a line through the points. Larger v_s 's do lead to larger values of D , however it is not clear from the graph if this can be said to be linear. While the value of $R^2 = 0.87$ is not negligible there are large residuals for each measurement. Additionally, in this case we have altered the value of v_s by changing a physical parameter of the system, namely D_p . If we were somehow able to hold v_s constant and change D_p one might well expect to measure different values for D . The only mechanism available to change v_s without altering D_p is to increase the saturation as we have seen above in [section 8.5.3](#). Other mechanisms to change v_s would involve changing the physical parameters (density and viscosity) of the fluid or the accelerating force (here gravity) but in the real world these cannot be changed.

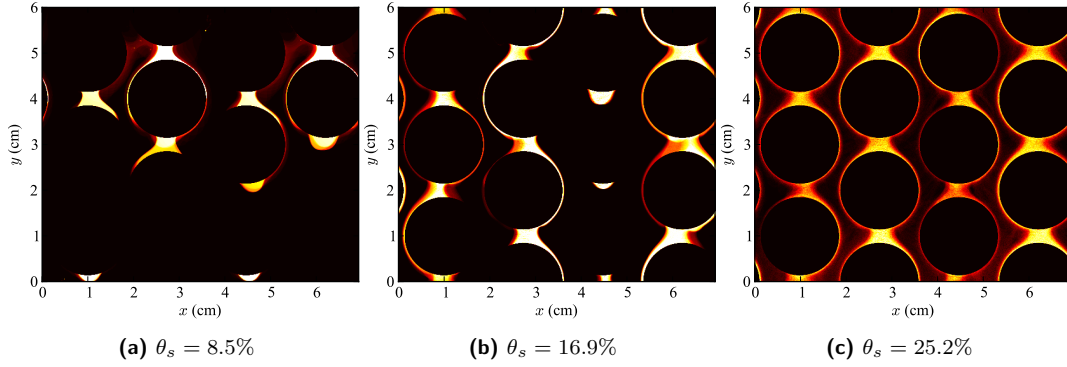


Figure 8.27: The 2D occupation histograms for the regular packing with an alternative gravity direction show what a large effect the system geometry can have on the flow. For low saturations rivulets form which are stable but can never bridge to neighbouring particles giving $D_{\text{radial}} = 0$.

8.6.2 Gravity direction for regular packing

The results presented up to this point were for a regular packing as shown in [figure 8.10](#) with gravity in the $-y$ direction. However another ‘natural’ orientation for the regular packing is where gravity is 30° from the vertical so that it lines up with a different line of symmetry of the packing as is shown in [figure 8.27](#). The final dispersion coefficients and Péclet numbers are shown in [figure 8.28](#). For this simulation setup, at low saturations, the fluid simply flows from one circle onto the circle directly below it and there is not enough fluid to bridge to another column of circles. This means that for low saturations we see effectively zero radial transport. Even at higher saturations, the radial transport is still very much suppressed in this case, as compared to the alternate orientation. Even though there is sufficient fluid to bridge between the columns of particles, the fluid prefers to flow directly down the relatively open channels as one might expect.

The lack of radial transport also has an effect on the axial transport. We can see from [figure 8.27b](#) that the fluid forms very stable, flowing, rivulets. In these rivulets the fluid is relatively well mixed meaning it has quite a low axial dispersion. This is because, at low saturations, the flowing rivulets have a small spatial extent. As before, the droplets necessarily act to enhance the axial dispersion, but they constitute a relatively smaller proportion of the fluid in the system and they are only present at the lowest flowing saturations. As the saturation increases the rivulets become larger and more unsteady, thereby increasing the inverse Péclet number by a combination of Taylor dispersion and the inherent randomness in the flow.

8.6.3 Irregular packing

The regular packings, while informative, display symmetries which are not present in real packed beds; even packed beds of monodisperse particles. For this reason irregular packings were generated using the procedure detailed in [section 4.3](#). Four different packings were generated for particles with a radius of 1 cm and with padding of 2 mm between the particles. The

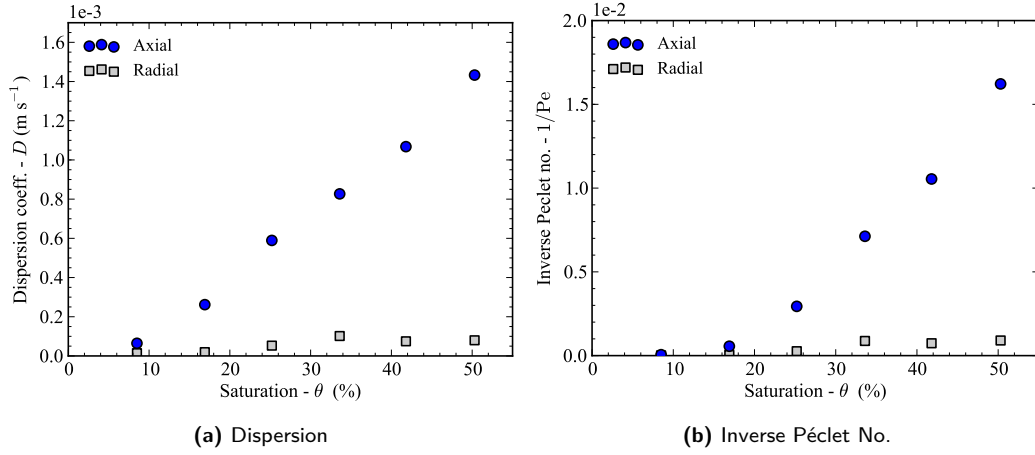


Figure 8.28: The dependence of the dispersion on the saturation is completely different for this case, with an alternate gravity direction, to the previous case with the same geometry. Here the droplets do not form at low saturations, instead the flow is confined to one particle column therefore we do not see the same enhancement of dispersion.

packing geometries can be seen in [figure 8.29](#) which shows the 2D occupation histograms for each geometry. To test the sensitivity of the flow to the exact packing geometry the results from simulations with a saturation of 20% are compared.

[Figure 8.30](#) shows the resulting seepage velocity for the four geometries. The error bars are calculated as the standard deviation of $v_s(t)$ throughout the course of the simulations. There is some variation in the seepage velocity outside the range of the error bars but this can easily be explained as the different geometries having slightly different porosities. Along side the seepage velocities the porosities, ϵ , are shown in light grey. There is a clear and strong correlation between ϵ and v_s . This will naturally have an effect on the dispersion measured for the different geometries, but plotting the dimensionless dispersivity against the dimensionless time allows us to compare the behaviour of the systems. This is shown in [figure 8.31](#). The dispersivity $\tilde{D}$ for the different geometries is very similar at short times which is to be expected since the dispersion is effectively Taylor. For larger times the dispersion coefficients remain similar but there is some difference. This difference, however, is primarily an effect of the limited sample size for large t . Since at the largest times we have only 1 sample for the displacements used to calculate D the resulting value is heavily dependent on the exact states of the systems. To counter this, one would need to run the simulation for a far longer time so that there were more samples for a given value of t . Unfortunately this represents an order of magnitude increase in computation time to either simulate longer or simulate more realisations of the same system.

8.7 Channelling

The interaction of the fluid flow, induced by gravity, with the solid boundaries in the system will act to spread the fluid throughout the domain. This is, however, countered by the action of capillarity. One large droplet has a lower surface area than multiple droplets adding to the

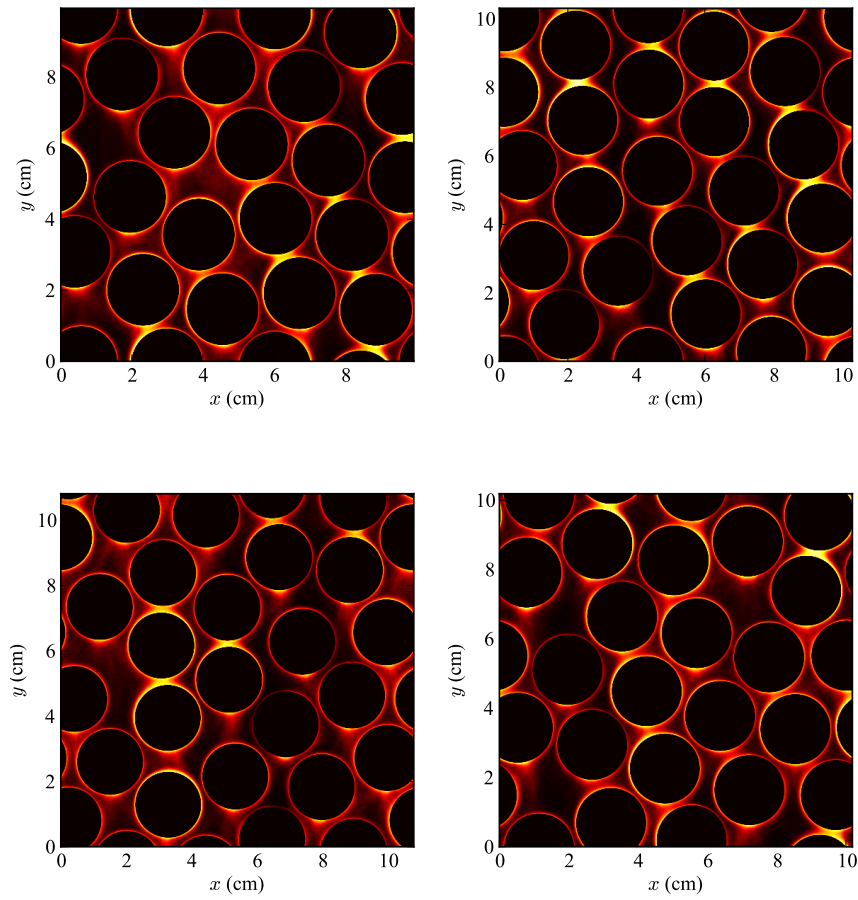


Figure 8.29: The 2D occupation histogram for the four different irregular geometries tested.

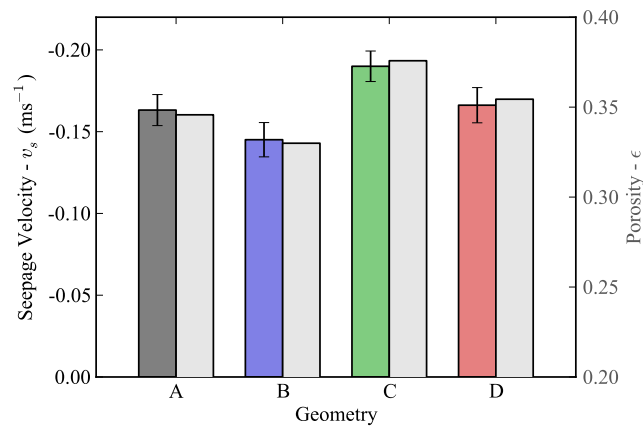


Figure 8.30: The seepage velocity for the four different geometries shows some variation but this can be explained due to the slight differences in porosity which arise from the packing procedure.

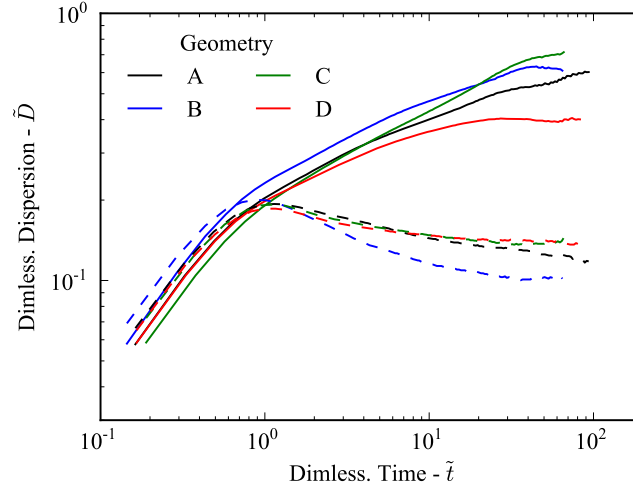


Figure 8.31: The axial dispersivity (solid) and radial dispersivity (dashed) for four different packings of 1 cm radius circles with 2 mm padding. The dispersivities are approximately equal for each packing (upto sampling noise at higher t) demonstrating that the dispersion is insensitive to the exact geometry used.

same volume therefore if droplets meet they will join and only the interaction with the solid boundary will cause them to separate again. It is partly this action which leads to the formation of preferential flow channels inside real heaps. In real heaps other factors at play help these form, such as mineral precipitation and blockage due to fines.

A vivid example of this action of capillary forces can be seen by considering flow in a system where the particle size is small, $D_p = 0.25$ cm. The two simulations shown in [figure 8.32](#) have exactly the same geometry and saturation ($\theta_s = 15\%$) and yet their behaviour is very different. The first was initialized using the droplet initial conditions discussed in [section 8.5.2](#) and second as a vertical strip in the centre of the domain. The first simulation is completely quiescent because the fluid droplets cannot touch and thereby join together to overcome the capillarity holding them to the particles. In the second simulation despite the interactions with the solid boundary spreading the fluid somewhat, capillarity is strong enough to hold the fluid droplets mostly together so that their combined size is enough to initiate and maintain flow. In larger simulations which are flowing, the capillarity can pull together droplets leading to the spontaneous formation of fast flow paths. Although this is a 2D example the same thing can happen in 3D. In this case the capillarity allows the fluid to form rivulets. Since these rivulets can be very stable once formed [\[53\]](#) there is likely to be less interaction between the rivulets and the droplets than in our model 2D systems. This would imply that the dispersion one would expect should be larger, and perhaps more non-Fickian, in three dimensions than that observed in our 2D simulations.

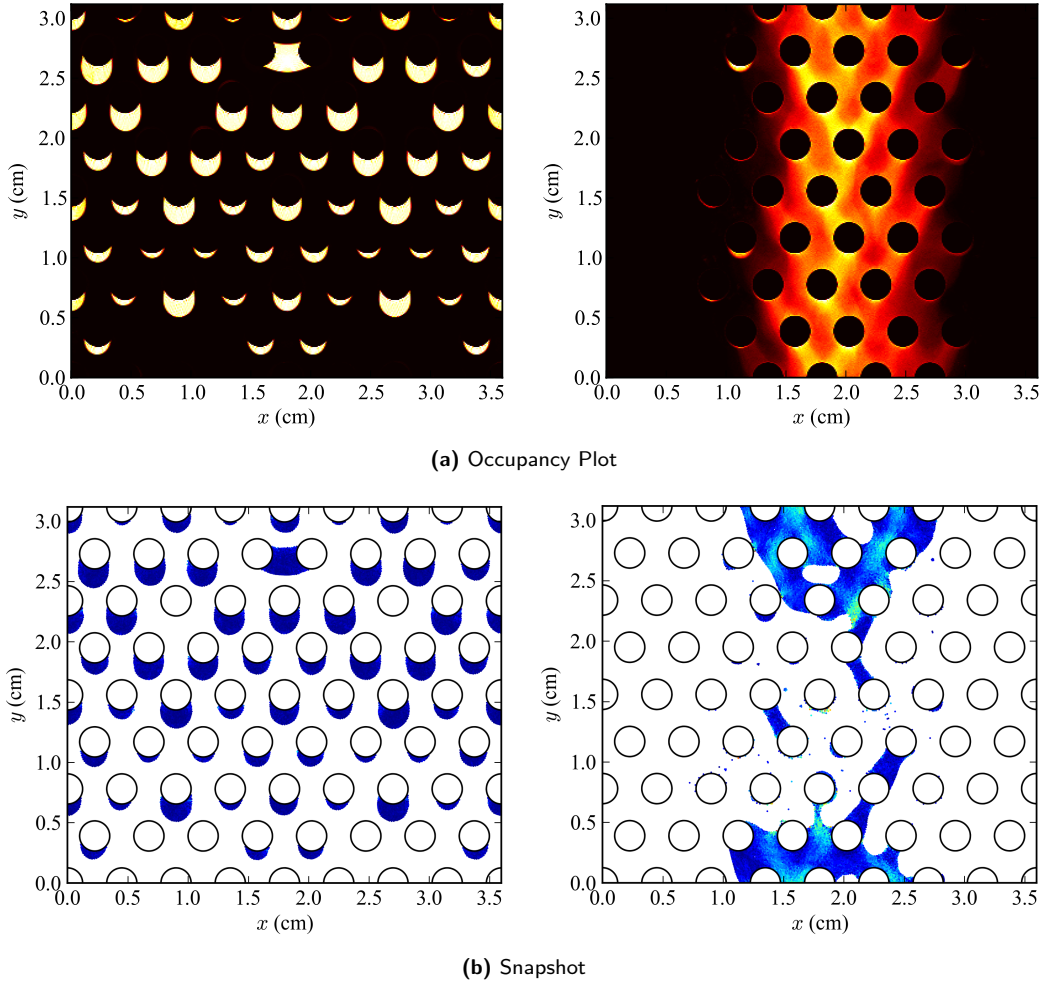


Figure 8.32: Two different starting conditions with the same saturations ($\theta_s = 15\%$) and the same particle size ($D_p = 0.25$ cm) show the powerful effect capillarity can have on the system. The first example is quiescent whereas in the second the preference for droplets to form larger droplets keeps the fluid in the domain together, rather than spreading throughout the domain, which allows gravity to cause the fluid to flow without having to overcome the capillarity.

8.8 Segregated Geometries

As discussed in [chapter 2](#) one of the features observed in real heaps is the segregation of particles based upon their size. This leads to flows in which one part of the domain can have a different characteristic length scale to another. To investigate the importance of this effect we generated a packing in which the average particle radius varies through the domain. Specifically in the periodic packing algorithm⁶ the particle radii were chosen such that for particle i the radius is

$$R_i = R_{\min} + (R_{\max} - R_{\min}) \frac{y_i}{L_y}. \quad (8.26)$$

⁶See [section 4.3](#).

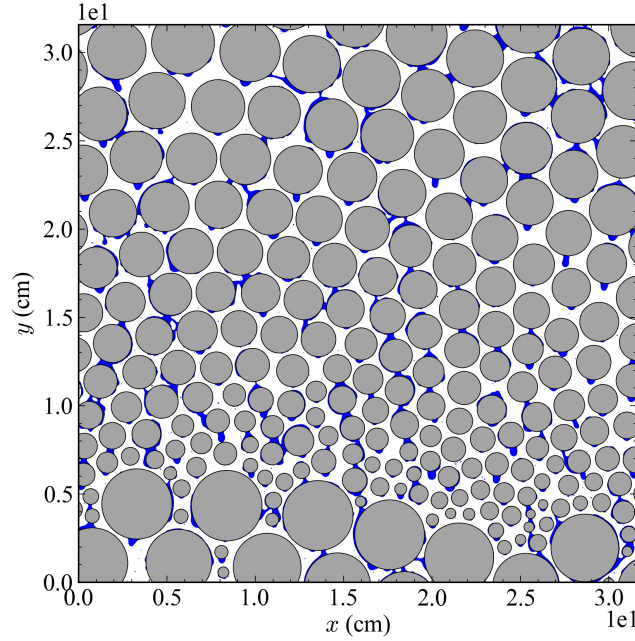


Figure 8.33: During the construction of heaps particles can segregate based on size. This geometry is representative of this process with generally decreasing particle size with depth. The (geometric) mean particle size is 0.7 cm.

This sets the radii based upon the *initial* y position of the particle but the particles can move past each other during the packing, as can be seen at the bottom of figure 8.33. This is of course similar to what would happen physically when a new rise is constructed on top of an existing heap. The final geometry, after packing, is shown in figure 8.33 it has period lengths $L_x = L_y = 31.575$ cm. This geometry was generated using $R_{\min} = 0.25$ cm and $R_{\max} = 2.00$ cm which is in line with the sizes seen in real heaps and as usual a padding width of 2 mm was left between the particles. For polydisperse packings it is typical to use the geometric, rather than the arithmetic, mean as the characteristic length [53]. Due to the way R_i is specified as linearly decreasing with y , the packing algorithm places more smaller particles into the domain than larger. This results in the geometric mean of the radii being somewhat lower than if we had radii uniformly distributed between $R_{\min}$ and $R_{\max}$. In this case we get $L = 0.71$ cm (2 s.f.)⁷.

Unlike our previous irregular packings, which had no particular preferred direction, the specification of the radii according to equation (8.26) clearly breaks this symmetry. For this reason the simulation was run with gravity at 5 different orientations; -90° , -45° , 0° , 45° and 90° . These five are sufficient because the symmetry under reflection in the y axis is maintained by this geometry. The simulations were carried out with a saturation of 10% and each used 980,000 SPH particles.

The first thing to observe is that the different orientations of the geometry do not lead to a

⁷See appendix A.9.

Table 8.2: The seepage velocity for different orientations.

Orientation	v_s (cm s ⁻¹)	Std. Dev. (cm s ⁻¹)
-90°	9.44	0.89
-45°	9.51	0.85
0°	9.83	0.91
45°	9.46	0.78
90°	9.67	1.14

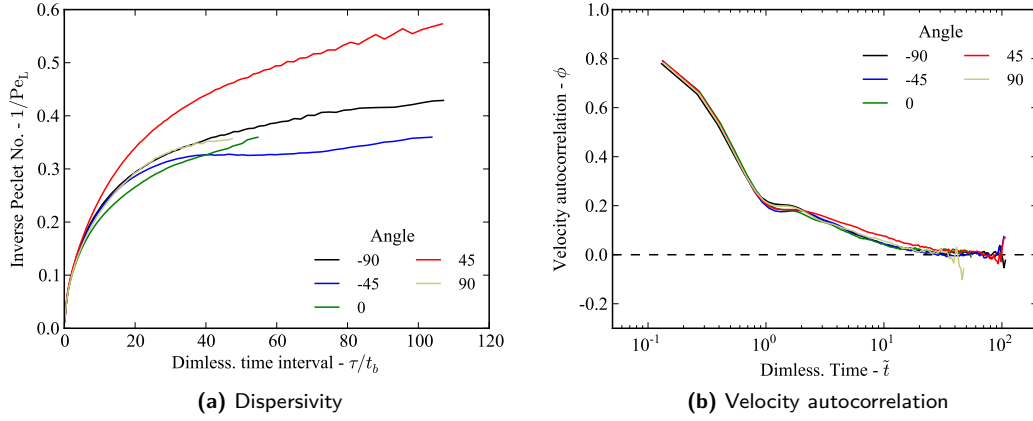


Figure 8.34: The dispersion coefficients for the different orientations are subject to significant noise but appear roughly similar. Comparing the velocity auto-correlation function - ϕ - confirms that across the whole system the orientation does have little effect on the observed transport.

noticeable difference in the seepage velocity (table 8.2). This is not surprising since even though the fluid is flowing through the geometry in different directions for each orientation, on average, the same amount of fluid can be held in each part of the domain (for different orientations). This means averaging the velocity over the whole domain will give us back approximately the same seepage velocity.

However, this hides an actual difference between the systems. Consider the pathline of an SPH particle in the system where the flow is parallel to the stratification. This SPH particle will (for short times at least) see the same average particle radius meaning its seepage velocity will stay approximately constant. However, the total ensemble of SPH particles in the system will see all of the different average particle radii (and thus local seepage velocities). For an SPH particle flowing parallel to the stratification the local particle radius (and hence seepage velocity) changes with time, but it does so for each fluid particle in the ensemble and thus the time averaged seepage velocity for a single SPH particle, in this case, will be the same as the overall ensemble seepage velocity. This is different for these two orientations.

Measuring the dispersion coefficients for the different orientations also seems to indicate that there is no real difference at short to medium times (figure 8.34a) but at long times the results are quite noisy. Examining the velocity auto correlation provides a better agreement between the systems as shown in figure 8.34b, but this too becomes quite noisy at long times ($\tilde{t} \gtrsim 10$).

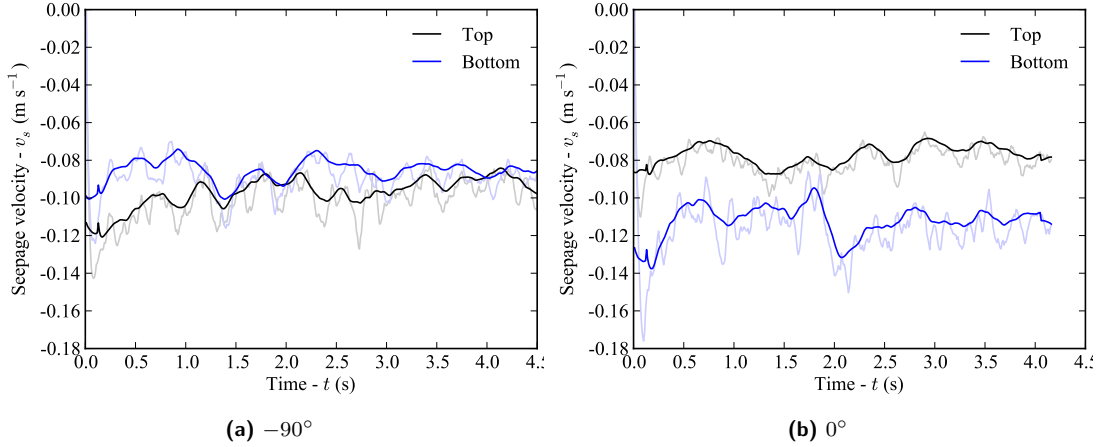


Figure 8.35: Considering the seepage velocity in two halves of the systems shows that the orientation of gravity can lead to spatial heterogeneities in the behaviour of the system.

While these results would suggest there is not much difference between the systems this belies the issue discussed above, namely the difference between time averaged quantities and ensemble averaged quantities. To emphasise this point we calculate the properties of the flow for pathlines starting in each half of the system; top and bottom respectively. In the systems where the flow is perpendicular to the stratification ($\pm 90^\circ$) each sub-dataset gives approximately equal values for the seepage velocity; 9.9 ± 1.0 cm and 8.6 ± 1.0 cm. The corresponding values for the parallel system (0°) are 11.2 ± 1.5 cm and 8.0 ± 0.8 cm. At long times the values for the two halves should converge to the ensemble average for the whole system due to mixing between the halves but this the differences seen here suggest that can take some time.

8.9 Conclusion

In this chapter we began by exploring the dispersion mechanisms of shear flow (Taylor dispersion) along with molecular diffusion. A new scheme was presented for simulating molecular diffusion atop the SPH flow. This scheme was verified by comparing the results of both a quiescent fluid and a 2D Poiseuille flow to the analytically known expected values for the measured dispersion. While molecular diffusion plays a key role in saturated creeping flows (without which there is theoretically no well defined dispersion coefficient) it was found that for the unsaturated and unsteady flows encountered in the interstices of heaps, that the hydrodynamic effects completely dominate the diffusion of substances. This is in part due to the scale of the hydrodynamic effects but also due to the discrete nature of the flows.

Next the mass transport within both regular and irregular 2D packings was studied by running a large number of SPH simulations of these systems. The boundaries represent a significant fraction of the simulation domains and thus the improvements for curved boundaries detailed in [chapter 5](#) are key for ensuring the accuracy of these simulations. The amount of dispersion was found to heavily depend on the saturation of the system. This is interestingly not just

a function of the increasing seepage velocity with saturation since the lowest dispersion was found for middling saturations of $\approx 50\%$. While a lower Reynolds number generally leads to a lower dispersion in these unsaturated flows the Reynolds number is coupled intrinsically to the saturation θ_s . For the lower values of θ_s the stationary droplets lead to an enhancement of the dispersion coefficient. This is confirmed by the velocity autocorrelation which persists for far longer times at lower θ_s than at higher saturations.

The simulations performed in this section also demonstrate that the precise geometry of the system (in terms of the particle spacing and sizes) has a large effect on the behaviours of both the resulting Reynolds numbers and the mass transport, as quantified by the dispersion coefficient. Apart from seeing large hysteresis in terms of the initial conditions we also observe that for particles below $\lesssim 0.5$ cm in diameter that the fluid is completely held up. This is in agreement with the generally accepted knowledge that the particles which constitute a heap should be ~ 2 cm in size. Further, for packings which display regular symmetries the alignment of these symmetries with the direction of gravity has a very large effect on how the transport phenomena behave. It is worth noting that the fluid can only be held up like this for unsaturated cases. If the beds were packed the fluid would be able to flow, albeit with decreasing seepage velocity.

A key observation we can make from our unsaturated simulations is that the dispersion coefficient takes several tens of dimensionless time units to approach its asymptotic value. While this means that over the height of a normal lift the fluid will have reached the Fickian regime, it implies that at distances of up to about half a metre the dispersion is non-Fickian. Indeed it appears to be super diffusive in this regime. This should definitely have a bearing on the sort of continuum models which are applied at the heap scale since this represents a sizeable fraction of the height of a heap and also how the results of smaller lab scale experiments are extrapolated for predicting heap scale experiments.

Chapter 9

Conclusions & Future Directions

The primary aim of this thesis was concerned with using the smoothed particle hydrodynamics (SPH) numerical method to studying mass transport and dispersion in the types of flows encountered in packed beds. In order to perform the necessary simulations several computational and numerical issues needed to be considered first. The handling of wall boundaries in SPH was studied and an improved method for simulating curved boundaries was presented as well as the inclusion of arbitrary shaped geometries using B-Splines. Additionally, the application of a new ‘boundary integral method’ for handling wall boundaries to surface tension and contact angle forces was investigated. Finally, a scheme for extracting mass transport behaviour from the SPH simulations, including the effects of molecular diffusion, was used to study the flows encountered in heaps.

The handling of curved boundaries using reflected ghost particles was improved through the addition of a volume-factor correction. Through the analysis of a Couette flow, for which the analytical solution is known, this correction was shown to improve the simulation accuracy for curved boundaries even at very high resolutions. This is vital for the accurate simulations performed in [chapter 8](#), in which the curved boundaries constituted a significant part of the simulation domains. The accurate handling of curved geometries is also important when including arbitrary shapes using the B-spline boundary framework which was developed. The computational and mathematical issues surrounding the inclusion of B-splines as wall boundaries in SPH were investigated. The efficacy of this formulation was tested in particular by the simulation of flow through a constricted pipe for which the expected flow behaviour was compared to a reference solution obtained using the Fluidity FEM code. Some discrepancy was seen between the FEM solution and the SPH solution, but this is not due to the boundaries, rather it is due to the discussed particle disorder, which is tied to the slightly compressible nature of the SPH formulation used in this thesis.

A new derivation and implementation of a recently presented method for handling arbitrary wall boundaries in SPH was discussed in some detail. Explicitly accounting for the surface integral terms, which arise close to the domain boundary, allows one to correct the kernel boundary deficiency using the correction factor - $\gamma(\mathbf{x})$. This correction manifests itself as a second sum in the SPH interpolation function, which runs over the boundary particles. The error associated with approximating the surface integral as a sum. In this fashion, was found to be small even for $B_f = 1$. This was achieved by comparing the value of the sum with the value given by a more advanced adaptive quadrature method. We demonstrated that this boundary correction can be applied to accurately reproduce the pressure and viscosity forces felt by the fluid. Indeed for a hydrostatic test case the initial fluid velocities, due to the mismatch between the curved boundary and the initially regular placement of the SPH particles, were found to be $\approx 40\%$ lower for the boundary integral method than for the reflected ghost particle boundaries.

The method, however, falls down when it comes to simulating surface tension forces. We have shown that the correction is only first order and is thus insufficient to correct for the surface tension forces which have twice the order of the smoothing kernel - W . This is an issue which has not been discussed in the literature. An attempted potential energy formulation for specifying contact angles, whereby the partially wetting fluid feels an additional acceleration into the wall, was found not to work satisfactorily for varying contact angles. A number of different functional forms were used for this acceleration and none were found to work. The factor which stops this method working properly is that by applying a force one disrupts the usual particle positioning near the boundary and more of the fluid tends to sit near the wall than should otherwise be the case.

It was found, however, that by using a different method to calculate the surface tension force, namely the curvature based method of [1], one can somewhat overcome this limitation of the boundary integral method. By adjusting the colours normals of particles that are near a boundary to point in the direction they would at equilibrium, one forces the droplet to adopt the correct contact angle. This comes at a cost however as the mismatch between the prescribed (equilibrium) colour gradient and the actual colour gradient causes an unphysical step in the droplet interface near the wall. Despite this drawback, this method works for a range of specified contact angles and is in any case superior to naïvely applying the boundary integral method to the CSF model.

Despite improvements to the handling of wall boundaries, the solutions provided by the SPH formulation can fail spectacularly if the numerical speed of sound - c_0 - is set too high. There are advantages to setting c_0 to a larger value, in that it reduces the compressibility of the simulated fluid and also appears to improve the spurious velocities seen at the interfaces between two or more phases, but care must be taken to choose the value appropriately. A suitable value according to [8] is taking c_0 to be 10 times the maximum velocity in the system. This works well for free surface flows where the fluid has room to move, but unfortunately in our closed systems even this can be too high since the velocity is potentially only high in a very small part of the domain, which leads to the fluid locking up elsewhere. An analysis of the forces at play in the system showed that this locking becomes an issue when the ratio of the pressure forces to the viscous forces becomes too skewed towards the pressure. This analysis showed that the error depends on certain physical parameters as $E \propto \rho_0 c_0^2 h / \mu s^{2d}$, which also demonstrates that as well as depending on the bulk modulus of the fluid ($\rho_0 c_0^2$), this effect is exacerbated by lower resolutions and smoothing factors.

In [chapter 7](#) preliminary work towards an improved method for including surface tension in SPH using Voronoi tessellation was presented. In this method a novel, yet simple, algorithm is employed to construct the Delaunay triangulation of the interface particles. Knowing the topology of the interface particles allows one to reconstruct an actual interface and thereby better calculate the surface tension forces and specifically account for the pressure discontinuity which occurs at fluid interfaces. This work is ongoing and has potential benefits in terms of triple point performance and alleviating some of the timestep constraints seen in unsaturated flows.

The scheme developed for simulating molecular diffusion is akin to a lazy random walker and therefore slightly different to other studies which either use an approximation of a continuous time random walk or solve the diffusion equation on top of their flows. It performs well on the validation cases of a quiescent fluid and Taylor dispersion in a pipe. The advantage of simulating the molecular diffusion in this way is that it ensures that it functions correctly even for unsaturated flows, has guaranteed conservation of mass, and can be extended so that the tracers are moving through an infinite periodic medium. The application of this scheme to a realistic unsaturated flow showed that there was no measurably different in the resultant dispersion coefficient. This agrees with the intuition that, at this scale, the inertial and shear effects are far more pronounced than molecular diffusion. This is, of course, only the case for the interstitial flow. For the flow within the ore particles themselves diffusion will play a large role.

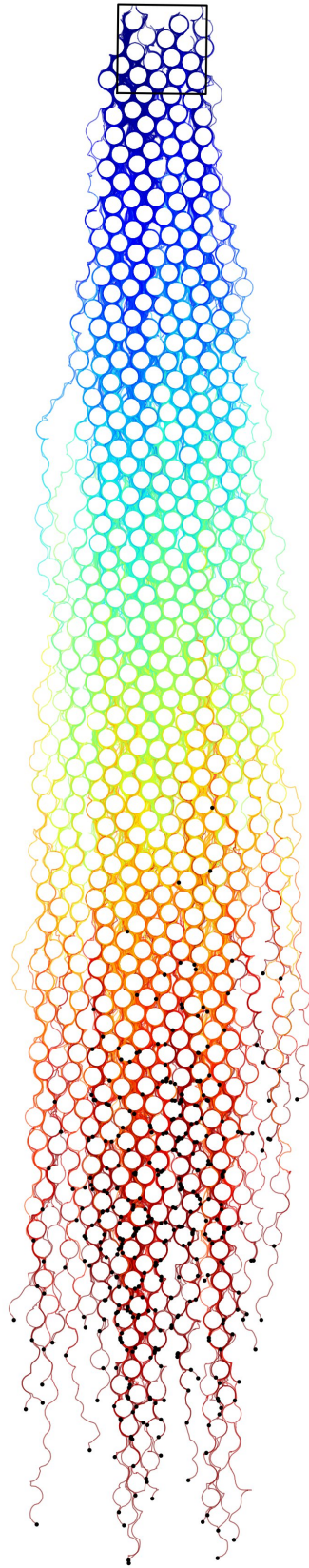
The simulations of regular packings highlight just how sensitive these systems can be to the initial conditions and the precise geometry of the particles. The behaviour of the inverse Péclet number with saturation was found to be completely different for two different orientations of the packing. For the first arrangement it was found that, at low saturations, the discrete nature of the flow acts to enhance the transport through the stagnation of fluid at capillary junctions or and hanging pendular droplets. Eventually these droplets are mixed back into the flow and the dispersion coefficient reaches a well defined value, but it seems, it can take several tens of dimensionless time units to get here. This is contrary to the usual modelling assumptions that the transport phenomena are Fickian even at very short times.

9.1 Future Work

The systems studied in this thesis, while informative, are mostly idealised, using circular ore particles and a single equilibrium contact angle. There remain many areas which could be fruitfully investigated in future work. The effects of particle shape could be studied using the boundary improvements and B-Spline boundaries. The tortuosity of flow paths is known to have an effect on mass transport so this is a logical area of investigation. A more thorough investigation of the effects of polydispersity would allow comparison of the simulation results to experimental studies and the behaviour seen in heaps. In addition, the observed channelling behaviour, which is caused by the action of capillarity, is potentially part of the reason for the establishment of preferential flow paths inside heaps. While hints of this were seen in the simulations carried out here, systematic study with larger scale simulations would be very useful for understanding the mechanisms at play and helping to develop theoretical models of such flow behaviour.

The two largest issues which lead to more physically accurate modelling of the transport within heaps are, firstly, performing 3D simulations, and secondly, accounting for contact angle hysteresis. The 3D simulations, unfortunately, are currently too computationally intensive to extract useful statistics from. Therefore a concentrated effort on developing a contact angle and surface tension formulation which does not require simulating the gas phase would be desirable. It is important to know what effect the finite size of our periodic domains has, since the domains

are necessarily smaller in 3D this effect would also need to be well quantified and understood. Finally, it is clear that the action of contact angle hysteresis will have a large effect on unsaturated systems such as packed beds. However, there is currently no method, which performs well, for including this effect into multiphase SPH simulations. This effect could potentially be included using the Voronoi surface tension formulation and therefore this is another area of potential future work.



Bibliography

- [1] Adami, S., Hu, X., and Adams, N. (2010). A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation. *J. Comput. Phys.*, 229(13):5011–5021.
- [2] AMCG (2011). Fluidity 4.1 Manual.
- [3] Antuono, M., Colagrossi, A., Touzé, D. L., and Monaghan, J. J. (2012). Conservation of circulation in SPH for 2D free-surface flows.
- [4] Barker, D. J., Brito-Parada, P. R., and Neethling, S. J. (2014). Application of B-Splines and Curved Geometries to Boundaries in SPH. *Int. J. Numer. Methods Fluids*, 76(1):51–68.
- [5] Barker, D. J. and Neethling, S. J. (2013). Dispersion of Unsaturated Unsteady Flow in 2D Periodic Packings. In Wills, B., editor, *Comput. Model.*, Falmouth, Cornwall, UK. Minerals Engineering International.
- [6] Barker, D. J., Parameswaran, G., and Neethling, S. J. (2012). SPH Simulation of Packed-beds and Columns Applied to Heap-leaching. In Solnordal, C., Liovic, P., Delaney, G., and Witt, P., editors, *9th Int. Conf. CFD Miner. Process Ind.*, number December, pages 1–6, Melbourne, Australia. CSIRO.
- [7] Bartlett, R. W. (1992). *Solution mining : leaching and fluid recovery of materials*. Gordon and Breach Science Publishers, Philadelphia, 2nd edition.
- [8] Becker, M. and Teschner, M. (2007). Weakly compressible SPH for free surface flows. In *Proc. 2007 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.*, pages 209–217, San Diego, California.
- [9] Belytschko, T., Krongauz, Y., Dolbow, J., and Gerlach, C. (1998). On the completeness of meshfree particle methods. *Int. J. Numer. Methods Eng.*, 43(5):785–819.
- [10] Bennett, C., McBride, D., Cross, M., and Gebhardt, J. (2012). A comprehensive model for copper sulphide heap leaching. *Hydrometallurgy*, 127-128:150–161.
- [11] Benson, D. a., Meerschaert, M. M., and Revielle, J. (2013). Fractional calculus in hydrologic modeling: A numerical perspective. *Adv. Water Resour.*, 51:479–497.
- [12] Bijeljic, B., Mostaghimi, P., and Blunt, M. J. (2011). Signature of Non-Fickian Solute Transport in Complex Heterogeneous Porous Media. *Phys. Rev. Lett.*, 107(20):204502.
- [13] Bijeljic, B., Raeini, A., Mostaghimi, P., and Blunt, M. J. (2013). Predictions of non-Fickian solute transport in different classes of porous media using direct simulation on pore-scale images. *Phys. Rev. E*, 87(1):13011.
- [14] Blunt, M. J., Bijeljic, B., Dong, H., Gharbi, O., Iglauer, S., Mostaghimi, P., Paluszny, A., and Pentland, C. (2013). Pore-scale imaging and modelling. *Adv. Water Resour.*, 51:197–216.
- [15] Bouffard, S. C. and Dixon, D. G. (2001). Investigative study into the hydrodynamics of heap leaching processes. *Metall. Mater. Trans. B*, 32(5):763–776.
- [16] Bouffard, S. C. and West-Sells, P. G. (2009). Hydrodynamic behavior of heap leach piles: Influence of testing scale and material properties. *Hydrometallurgy*, 98(1-2):136–142.
- [17] Brackbill, J. U. and Kothe, D. B. (1996). *Dynamical modeling of surface tension*, pages 693–700.
- [18] Brackbill, J. U., Kothe, D. B., and Zemach, C. (1992). A Continuum Method for modeling Surface Tension. *J. Comput. Phys.*, 100:335–354.

- [19] Breinlinger, T., Polfer, P., Hashibon, A., and Kraft, T. (2013). Surface Tension and Wetting Effects with Smoothed Particle Hydrodynamics. *J. Comput. Phys.*, 243:14–27.
- [20] Cai, L., Ferrier, R. J., Qingyang, L., Mostaghimi, P., Gorman, G. J., and Neethling, S. J. (2014). Heap Leaching Simulation: Beyond Shrinking Core Models. In Johansen, S. T., editor, *10th Int. Conf. CFD Oil Gas, Metall. Process Ind.*, number June, pages 1–7, Trondheim, Norway. SINTEF.
- [21] Cariaga, E., Concha, F., and Sepulveda, M. (2005). Flow through porous media with applications to heap leaching of copper ores. *Chem. Eng. J.*, 111(2-3):151–165.
- [22] Cartwright, A., Stamatellos, D., and Whitworth, A. P. (2009). The effect of Poisson noise on SPH calculations. *Mon. Not. R. Astron. Soc.*, 395(4):2373–2380.
- [23] Chen, Z. and Ewing, R. (1997). Fully discrete finite element analysis of multiphase flow in groundwater hydrology. *SIAM J. Numer. Anal.*, 34(6):2228–2253.
- [24] Colagrossi, A. (2003). Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *J. Comput. Phys.*, 191(2):448–475.
- [25] Cox, M. (1971). The Numerical Evaluation of B-Splines. *IMA J. Appl. Math.*, 10(2):134–149.
- [26] Cross, M., Bennett, C., Croft, T., McBride, D., and Gebhardt, J. (2006). Computational modeling of reactive multi-phase flows in porous media: Applications to metals extraction and environmental recovery processes. *Miner. Eng.*, 19(10):1098–1108.
- [27] Cummins, S. J., Silvester, T. B., and Cleary, P. W. (2012). Three-dimensional wave impact on a rigid structure using smoothed particle hydrodynamics. *Int. J. Numer. Methods Fluids*, 68(12):1471–1496.
- [28] Das, R. and Cleary, P. W. (2010). Effect of rock shapes on brittle fracture using Smoothed Particle Hydrodynamics. *Theor. Appl. Fract. Mech.*, 53(1):47–60.
- [29] de Andrade Lima, L. (2006). Liquid axial dispersion and holdup in column leaching. *Miner. Eng.*, 19(1):37–47.
- [30] de Arcangelis, L., Koplik, J., Redner, S., and Wilkinson, D. (1986). Hydrodynamic Dispersion in Network Models of Porous Media. *Phys. Rev. Lett.*, 57(8):996–999.
- [31] de Boor, C. (1972). On Calculating With B-Splines. *J. Approx. Theory*, 6:50–62.
- [32] Delgado, J. M. P. Q. (2005). A critical review of dispersion in packed beds. *Heat Mass Transf.*, 42(4):279–310.
- [33] Desbrun, M. (1996). Smoothed Particles : A new paradigm for animating highly deformable bodies. *Computer (Long. Beach. Calif.)*, pages 61–76.
- [34] Dixon, D. G. and Hendrix, J. L. (1993). A mathematical model for heap leaching of one or more solid reactants from porous ore pellets. *Metall. Trans. B*, 24(6):1087–1102.
- [35] Eriksson, N. and Destouni, G. (1997). Combined effects of dissolution kinetics, secondary mineral precipitation, and preferential flow on copper leaching from mining waste rock. *Water Resour. Res.*, 33(3):471–483.
- [36] Feldman, J. and Bonet, J. (2010). Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *Int. J. Numer. Methods Eng.*, 72(3):295–324.

- [37] Ferrand, M., Laurence, D. R., Rogers, B. D., Violeau, D., and Kassiotis, C. (2012). Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *Int. J. Numer. Methods Fluids*.
- [38] Ferrari, A. (2010). SPH simulation of free surface flow over a sharp-crested weir. *Adv. Water Resour.*, 33(3):270–276.
- [39] Flekkøy, E., Coveney, P., and De Fabritiis, G. (2000). Foundations of dissipative particle dynamics. *Phys. Rev. E*, 62(2):2140.
- [40] Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. Astron. Soc.*, 181(181):375–389.
- [41] Griebel, M., Knappek, S., and Zumbusch, G. (2007). *Numerical Simulation in Molecular Dynamics*, volume 5 of *Texts in Computational Science and Engineering*. Springer.
- [42] Griebel, M. and Zaspel, P. (2010). A multi-GPU accelerated solver for the three-dimensional two-phase incompressible Navier-Stokes equations. *Comput. Sci. Res. Dev.*, 25(1-2):65–73.
- [43] Guillon, V., Fleury, M., Bauer, D., and Neel, M. C. (2013). Superdispersion in homogeneous unsaturated porous media using NMR propagators. *Phys. Rev. E*, 87(4):043007.
- [44] Haga, D., Niibori, Y., and Chida, T. (1999). Hydrodynamic dispersion and mass transfer in unsaturated flow. *Water Resour. Res.*, 35(4):1065–1077.
- [45] Haga, D., Niibori, Y., and Chida, T. (2000). Tracer responses in gas–liquid two-phase flow through porous media. *Proc. World Geotherm. ...*, pages 2591–2596.
- [46] Ham, D. A., Farrell, P. E., Gorman, G. J., Maddison, J. R., Wilson, C. R., Kramer, S. C., Shipton, J., and Collins, G. S. (2009). Model Development Spud 1.0 : generalising and automating the user interfaces of scientific computer models. *Geosci. Model Dev.*, 2:33–42.
- [47] Harada, T., Koshizuka, S., and Kawaguchi, Y. (2007). Smoothed particle hydrodynamics in complex shapes. In Sbert, M., editor, *Proc spring Conf. Comput. Graph.*, pages 26–28, Budmerice, Slovakia.
- [48] Hu, X. and Adams, N. (2007). An incompressible multi-phase SPH method. *J. Comput. Phys.*, 227(1):264–278.
- [49] Hu, X. Y. and Adams, N. A. (2006). A multi-phase SPH method for macroscopic and mesoscopic flows. *J. Comput. Phys.*, 213(2):844–861.
- [50] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, 9(3):90–95.
- [51] Ilankoon, I., Cole, K., and Neethling, S. (2013). Measuring hydrodynamic dispersion coefficients in unsaturated packed beds: Comparison of PEPT with conventional tracer tests. *Chem. Eng. Sci.*, 89:152–157.
- [52] Ilankoon, I. and Neethling, S. (2012). Hysteresis in unsaturated flow in packed beds and heaps. *Miner. Eng.*, 35:1–8.
- [53] Ilankoon, I. M. S. K. (2012). *Hydrodynamics of Unsaturated Particle Beds Pertaining to Heap Leaching*. Phd thesis, Imperial College London.
- [54] Jergensen, G. V., editor (1999). *Copper Leaching, Solvent Extraction, and Electrowinning Technology*. SME.

- [55] Kappes, D. (2007). Precious metal heap leach design and practice. Technical report, Kappes, Cassiday & Associates, Reno, Nevada.
- [56] Kelager, M. (2006). Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics.
- [57] Kesler, S. E. (2007). Mineral supply and demand into the 21st century. In *Work. Depos. Model. Miner. Resour. Assessment, Sustain. Dev.*, pages 55–62.
- [58] Kitsionas, S. and Whitworth, A. P. (2008). Smoothed Particle Hydrodynamics with particle splitting, applied to self-gravitating collapse. *Mon. Not. R. Astron. Soc.*, (February):1–9.
- [59] Koshizuka, S., Nobe, A., and Oka, Y. (1998). Numerical analysis of breaking waves using the moving particle semi-implicit method. *Int. J. Numer. Methods Fluids*, 26(7):751–769.
- [60] Krog, O. y. and Elster, A. (2012). Fast GPU-Based Fluid Simulations Using SPH. In Jónasson, K., editor, *Appl. Parallel Sci. Comput.*, volume 7134 of *Lecture Notes in Computer Science*, pages 98–109. Springer Berlin Heidelberg.
- [61] Krog, O. y. E. (2010). GPU-based Real-Time Snow Avalanche Simulations. (June).
- [62] Kubo, R. (1966). The fluctuation-dissipation theorem. *Reports Prog. Physics*1, 29:255–284.
- [63] Kulasegaram, S., Bonet, J., Lewis, R. W., and Profit, M. (2004). A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications. *Comput. Mech.*, 33(4):316–325.
- [64] Lafaurie, B., Nardone, C., Scardovelli, R., Zaleski, S., and Zanetti, G. (1994). Modelling merging and fragmentation in multiphase flows with SURFER. *J. Comput. Phys.*, 113(1):134–147.
- [65] Leahy, M. J., Schwarz, M. P., and Davidson, M. R. (2003). AN AIR SPARGING CFD MODEL FOR HEAP BIOLEACHING OF COPPER-SULPHIDE. In *Third Int. Conf. CFD Miner. Process Ind.*, number December, pages 581–586.
- [66] Li, S. and Liu, W. K. (2004). *Meshfree particle methods*, volume 25. Springer Verlag.
- [67] Lind, S., Xu, R., Stansby, P., and Rogers, B. (2011). Incompressible Smoothed Particle Hydrodynamics for Free-Surface Flows: A Generalised Diffusion-Based Algorithm for Stability and Validations for Impulsive Flows and Propagating Waves. *J. Comput. Phys.*
- [68] Liu, M. (2003). Constructing smoothing functions in smoothed particle hydrodynamics with applications. *J. Comput. Appl. Math.*, 155(2):263–284.
- [69] Liu, M. and Liu, G. (2006). Restoring particle consistency in smoothed particle hydrodynamics. *Appl. Numer. Math.*, 56(1):19–36.
- [70] Liu, M. B. and Liu, G. R. (2010). Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Arch. Comput. Methods Eng.*, 17(1):25–76.
- [71] Lowe, C. and Frenkel, D. (1996). Do Hydrodynamic Dispersion Coefficients Exist? *Phys. Rev. Lett.*, 77(22):4552–4555.
- [72] Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *Astron. J.*, 82(12):1013–1024.
- [73] Maiti, R., Khanna, R., and Nigam, K. D. P. (2006). Hysteresis in Trickle Bed Reactors: A Review. *Ind. Eng. Chem. Res.*, 45(15):5185–5198.

- [74] McBride, D., Cross, M., Croft, N., Bennett, C., and Gebhardt, J. (2006). Computational modelling of variably saturated flow in porous media with complex three-dimensional geometries. *Int. J. Numer. Methods Fluids*, 50(August 2005):1085–1117.
- [75] McClelland, G. E., Pool, D. L., and Eisele, J. A. (1983). Agglomeration-Heap Leaching Operations in the Precious Metals Industry. *Bur. Mines Informait. Circ.*, 8945.
- [76] Monaghan, J. J. (2000). SPH without a Tensile Instability. *J. Comput. Phys.*, 159(2):290–311.
- [77] Monaghan, J. J. and Lattanzio, J. C. (1985). A refined particle method for astrophysical problems. *Astron. Astrophys.*, 149:135–143.
- [78] Morris, J. P. (1996). *Analysis of Smoothed Particle Hydrodynamics with Applications*. PhD thesis, Monash University.
- [79] Mostaghimi, P., Ilankoon, I. S. K., and Neethling, S. J. (2014). Use of mesh adaptivity in simulation of flow in packed beds – A case study. *Miner. Eng.*, 63:157–163.
- [80] Müller, M., Charypar, D., and Gross, M. (2003). Particle-Based Fluid Simulation for Interactive Applications. *Fluid Dyn.*, pages 154–159.
- [81] Müller, M., Schirm, S., and Teschner, M. (2004). Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics. *Technol. Heal. care Off. J. Eur. Soc. Eng. Med.*, 12(1):25–31.
- [82] Oger, G., Doring, M., Alessandrini, B., and Ferrant, P. (2006). Two-dimensional SPH simulations of wedge water entries. *J. Comput. Phys.*, 213(2):803–822.
- [83] Omidvar, P., Stansby, P. K., and Rogers, B. D. (2012). SPH for 3D floating bodies using variable mass particle distribution. *Int. J. Numer. Methods Fluids*, 72(4):427–452.
- [84] Ovaysi, S. and Piri, M. (2010). Direct pore-level modeling of incompressible fluid flow in porous media. *J. Comput. Phys.*, 229(19):7456–7476.
- [85] Owen, J., Villumsen, J., Shapiro, P., and Martel, H. (1998). Adaptive Smoothed Particle Hydrodynamics: Methodology. II. *Astrophys. J. Suppl. Ser.*, 116:155.
- [86] Pereira, G. and Prakash, M. (2011). SPH modelling of fluid at the grain level in a porous medium. *Appl. Math. Model.*, 35(4):1–6.
- [87] Raeini, A. Q. (2013). *Modelling multiphase flow through micro-CT images of the pore space*. PhD thesis, Imperial College London.
- [88] Rogers, D. F. (2001). *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann, San Francisco; London.
- [89] Serrano, M. and Español, P. (2001). Thermodynamically consistent mesoscopic fluid particle model. *Phys. Rev. E*, 64(4):1–18.
- [90] Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proc. 1968 23rd ACM Natl. Conf.*, ACM '68, pages 517–524, New York, NY, USA. ACM.
- [91] Sigalotti, L. D. G., Klapp, J., Sira, E., Melean, Y., and Hasmy, A. (2003). SPH simulations of time-dependent Poiseuille flow at low Reynolds numbers. *J. Comput. Phys.*, 191(2):622–638.

- [92] Sousa, A. C. M. and Jiang, F. (2007). SPH as an inverse numerical tool for the prediction of diffusive properties in porous media. *Mater. Sci. Forum*, 553(Diffusion in Solids and Liquids):171–189.
- [93] Springel, V. (2010). Smoothed Particle Hydrodynamics in Astrophysics. *Annu. Rev. Astron. Astrophys.*, 48:391–430.
- [94] Suzuki, Y., Koshizuka, S., and Oka, Y. (2007). Hamiltonian moving-particle semi-implicit (HMPS) method for incompressible fluid flows. *Comput. Methods Appl. Mech. Eng.*, 196(29-30):2876–2894.
- [95] Swegle, J. W., Hicks, D. L., and Attaway, S. W. (1995). Smoothed Particle Hydrodynamics Stability Analysis. *J. Comput. Phys.*, 116(1):123–134.
- [96] Tompson, A. (1988). On a new functional form for the dispersive flux in porous media. *Water Resour. Res.*, 24(11):1939–1947.
- [97] Tompson, A. F. B. and Gray, W. G. (1986a). A Second-Order Approach for the Modeling of Dispersive Transport in Porous Media 1. Theoretical Development. *Water Resour. Res.*, 22(5):591–599.
- [98] Tompson, A. F. B. and Gray, W. G. (1986b). A second-order approach for the modeling of dispersive transport in porous media 2. Application to Solute Motion in Pipes and Capillary Tubes. *Water Resour. Res.*, 22(5):601–614.
- [99] Tompson, A. F. B. and Gray, W. G. (1986c). A second-order approach for the modeling of dispersive transport in porous media: 3. Application to two porous media problems. *Water Resour. Res.*, 22(13):1959–1971.
- [100] Wendl, M. C. (1999). General solution for the Couette flow profile. *Phys. Rev. E. Stat. Phys. Plasmas. Fluids. Relat. Interdiscip. Topics*, 60(5 Pt B):6192–4.
- [101] Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(1):389–396.
- [102] Wills, B. A. (2006). *Will's Mineral Processing Technology*. Elsevier, 7 edition.
- [103] Wu, a., Yin, S., Qin, W., Liu, J., and Qiu, G. (2009). The effect of preferential flow on extraction and surface morphology of copper sulphides during heap leaching. *Hydrometallurgy*, 95(1-2):76–81.
- [104] Yusuf, R. (1984). *Liquid Flow Characteristics in Heap and Dump Leaching*. Msc. thesis, University of New South Wales, Australia.
- [105] Zhu, Y. and Fox, P. J. (2002). Simulation of Pore-Scale Dispersion in Periodic Porous Media Using Smoothed Particle Hydrodynamics. *J. Comput. Phys.*, 182(2):622–645.

Appendix A

Derivations

A.1 Derivation of Expected Speed Up - S_p

For 3D the volume of the whole system is L^3 . Let each process be a cube with side l , the volume of subdomain is $l^3 = L^3/N_p$ where N_p is the number of processors. Because the data transferred is only that at the surface of the subdomain, the time spent transferring is $T_t \sim l^2$ but we have from above

$$l^2 = (l^3)^{\frac{2}{3}} = \left(\frac{L^3}{N_p}\right)^{\frac{2}{3}} \sim N_p^{-\frac{2}{3}}.$$

The computation time for each processor is inversely proportional to the number of processors, $T_c \propto 1/N_p$, meaning the total step time (computation + transfer) is given by

$$T = \frac{A}{N_p} + \frac{B}{N_p^{\frac{2}{3}}},$$

for some constants A and B . The speed up $S_p(N_p)$ is defined to be the ratio $T(1)/T(N_p)$ which substituting from above and rearranging gives

$$S_p = \frac{N_p}{A' + B' \sqrt[3]{N_p}}$$

where $A' = A/(A + B)$ and $B' = B/(A + B)$. Thus the scaling in the limit

$$\lim_{N_p \rightarrow \infty} \frac{N_p}{(A + B \sqrt[3]{N_p})} \cdot \frac{1}{N_p^\beta} = \text{const.},$$

gives us $\beta = 2/3$ hence $S_p \sim N_p^{2/3}$. $\square$

The different results for slices and pillars come from the fact that the surface area of the subdomains scales as l^1 and l^2 respectively.

A.2 Derivation of CSF Model

The interface force due to surface tension is given by

$$\mathbf{f} = -\alpha\kappa\hat{\mathbf{n}} \tag{A.1}$$

where κ is the mean curvature and can be expressed $\nabla \cdot \hat{\mathbf{n}}$. In the CSF model we smooth slightly the interface and have the force instead become a volume force which acts over a small width either side of the interface. The force must go to zero away from the interface and the weighting must integrate to unity. The natural candidate then is to weight the force by the

gradient of the colour function $|\nabla C|$ which has the desired properties, thus in the CSF model we have

$$\mathbf{f} = -\alpha \hat{\mathbf{n}} \nabla \cdot \hat{\mathbf{n}} |\nabla C|. \quad (\text{A.2})$$

Consider the expression

$$\nabla \cdot (|\nabla C| \hat{\mathbf{n}} \otimes \hat{\mathbf{n}}), \quad (\text{A.3})$$

expanding this using the product rule gives us

$$\nabla \cdot (|\nabla C| \hat{\mathbf{n}} \otimes \hat{\mathbf{n}}) = \hat{\mathbf{n}} \nabla \cdot (|\nabla C| \hat{\mathbf{n}}) + (|\nabla C| \hat{\mathbf{n}} \cdot \nabla) \hat{\mathbf{n}} \quad (\text{A.4})$$

further expanding the divergence in the first term and bringing $|\nabla C|$ out of the brackets in the second gives

$$= \hat{\mathbf{n}} (|\nabla C| \nabla \cdot \hat{\mathbf{n}} + \hat{\mathbf{n}} \cdot \nabla (|\nabla C|)) + |\nabla C| (\hat{\mathbf{n}} \cdot \nabla) \hat{\mathbf{n}} \quad (\text{A.5})$$

$$= \hat{\mathbf{n}} |\nabla C| \nabla \cdot \hat{\mathbf{n}} + \hat{\mathbf{n}} (\hat{\mathbf{n}} \cdot \nabla (|\nabla C|)) + |\nabla C| (\hat{\mathbf{n}} \cdot \nabla) \hat{\mathbf{n}} \quad (\text{A.6})$$

We can see the first term is exactly the force we are after, so we rearrange and bring this term on it own

$$-\hat{\mathbf{n}} |\nabla C| \nabla \cdot \hat{\mathbf{n}} = \hat{\mathbf{n}} (\hat{\mathbf{n}} \cdot \nabla (|\nabla C|)) + |\nabla C| (\hat{\mathbf{n}} \cdot \nabla) \hat{\mathbf{n}} - \nabla \cdot (|\nabla C| \hat{\mathbf{n}} \otimes \hat{\mathbf{n}}). \quad (\text{A.7})$$

To proceed further we utilise the relationships between ∇C and $\hat{\mathbf{n}}$. The first term on the right hand side of [equation \(A.7\)](#) is actually just equal to $\nabla (|\nabla C|)$. This is because $\nabla (|\nabla C|)$ necessarily points in the same direction as $\hat{\mathbf{n}}$ so we are in effect projecting onto $\hat{\mathbf{n}}$ then remultiplying by $\hat{\mathbf{n}}$ giving us back to the original quantity. Next we notice that

$$(\hat{\mathbf{n}} \cdot \nabla) \hat{\mathbf{n}} \equiv 0$$

thereby eliminating the second term from [equation \(A.7\)](#). This is only the case because of the special form of the normals. In effect, the normals point in the shortest direction towards the interface, thus if we move in this direction for some distance then the most direct vector to the interface (i.e. the normal) has not changed. This finally gives us that

$$\begin{aligned} -\hat{\mathbf{n}} |\nabla C| \nabla \cdot \hat{\mathbf{n}} &= \nabla (|\nabla C|) - \nabla \cdot (|\nabla C| \hat{\mathbf{n}} \otimes \hat{\mathbf{n}}), \\ &= \nabla \cdot \underbrace{(|\nabla C| \mathbf{I} - |\nabla C| \hat{\mathbf{n}} \otimes \hat{\mathbf{n}})}_{=\Pi}. \end{aligned} \quad (\text{A.8})$$

This differs from [equation \(3.50\)](#) by a factor of $1/d$ in front of the identity matrix. It is explained in [\[49\]](#) that this is to correct for a spurious pressure which would otherwise arise using this formulation. Because the normals are unit vectors, $\mathbf{I} - \hat{\mathbf{n}} \otimes \hat{\mathbf{n}}$ has a trace of $d - 1$. Thus when taking the divergence there is a pressure type force which arises (c.f. non deviatoric part of the stress tensor). By dividing the first factor in [equation \(A.8\)](#) by d we are ensuring that $\text{Tr}(\Pi) \equiv 0$.

A.3 Derivation of Local Pressure Force

We wish to calculate the local force due to pressure felt by displacing an SPH particle. Imagine a particle i whose position we update as $\mathbf{r}_i = (r_x + \varepsilon, r_y)^T$. Imagine the particle to sit in a regular cubic lattice in 2D, then the only forces which will be unbalanced will be the due to the particles in the x -direction which we denote as particles $i-1$ and $i+1$. We wish to know what form this force takes to understand its dependence on c_0 , ρ_0 , μ and h .

The force between two particles i and j due to pressure is given by

$$\mathbf{f}_{ij} = \frac{1}{V} \left(\frac{u(\rho_i^*)}{\sigma_i^2} + \frac{u(\rho_j^*)}{\sigma_j^2} \right) \nabla W_{ij} \quad (\text{A.9})$$

thus the x component of the force due to the two neighbouring particles $i+1$ and $i-1$ is

$$f_x = \frac{1}{V} \left[\left(\frac{u(\rho_i^*)}{\sigma_i^2} + \frac{u(\rho_{i-1}^*)}{\sigma_{i-1}^2} \right) \nabla W_{i,i-1} + \left(\frac{u(\rho_i^*)}{\sigma_i^2} + \frac{u(\rho_{i+1}^*)}{\sigma_{i+1}^2} \right) \nabla W_{i,i+1} \right] \quad (\text{A.10})$$

In order to simplify equation (A.10) we note that

$$\sigma_i = \frac{\rho_i^*}{V} \quad (\text{A.11})$$

this then gives

$$f_x = V \left[\left(\frac{u(\rho_i^*)}{(\rho_i^*)^2} + \frac{u(\rho_{i-1}^*)}{(\rho_{i-1}^*)^2} \right) \nabla W_{i,i-1} + \left(\frac{u(\rho_i^*)}{(\rho_i^*)^2} + \frac{u(\rho_{i+1}^*)}{(\rho_{i+1}^*)^2} \right) \nabla W_{i,i+1} \right] \quad (\text{A.12})$$

next we note that

$$\nabla W_{ij} = \frac{\partial W(r)}{\partial r} \bigg|_{r=|\mathbf{r}_i - \mathbf{r}_j|} \hat{\mathbf{e}}_{ij} = -\nabla W_{ji} \quad (\text{A.13})$$

and that because we are considering a small displacement ε from an initially regular grid that the distances involved are $r_{i,i-1} = dx + \varepsilon$ and $r_{i,i+1} = dx - \varepsilon$ with $\varepsilon \ll dx$ giving $r_{i,i-1} \approx r_{i,i+1} \approx dx$. This gives $\nabla W_{i,i-1} \approx -\nabla W_{i,i+1} \approx -W'(dx)$ where $W' = \frac{\partial W(r)}{\partial r}$. Plugging these back into equation (A.12) yields

$$f_x \approx -V \left(\frac{u(\rho_{i+1}^*)}{(\rho_{i+1}^*)^2} - \frac{u(\rho_{i-1}^*)}{(\rho_{i-1}^*)^2} \right) W'(dx) \quad (\text{A.14})$$

next taking $u(\rho^*)$ as equation (3.28) gives

$$f_x \approx -\frac{V\rho_0 c_0^2}{\gamma} \left(\frac{(\rho_{i+1}^*)^\gamma - 1}{(\rho_{i+1}^*)^2} - \frac{(\rho_{i-1}^*)^\gamma - 1}{(\rho_{i-1}^*)^2} \right) W'(dx) \quad (\text{A.15})$$

In order to make progress we need to know next how ρ_{i-1}^* and ρ_{i+1}^* vary with ε . We expect that as we increase ε (move i to the right) that ρ_{i+1}^* should increase and vice versa ρ_{i-1}^* should decrease. In reality the form of $\rho_j^*(\varepsilon)$ will depend in some non-linear way on the form of W chosen however since we are dealing with small displacements we make the assumption the densities will vary linearly with ρ_{i-1}^* and ρ_{i+1}^* varying as $1 - k\varepsilon$ and $1 + k\varepsilon$ respectively.

the Taylor series expansion (up to first order) of $W(r)$ around $r = dx$ is given by

$$W(dx + \varepsilon) = W(dx) + \varepsilon W'(dx) + O(\varepsilon^2) \quad (\text{A.16})$$

Since we have

$$\rho_{i\pm 1}^* = \frac{m}{\rho_0} \left(\left(\sum_{j \neq i} W_{ij} \right) + W_{i\pm 1, i}(dx + \varepsilon) \right) \quad (\text{A.17})$$

we can then plug in the series expansion [equation \(A.16\)](#) giving

$$\rho_{i\pm 1}^* = \frac{m}{\rho_0} \left(\left(\sum_{j \neq i} W_{ij} \right) + W(dx) \right) + \frac{m}{\rho_0} (W'(dx)\varepsilon + O(\varepsilon^2)) \quad (\text{A.18})$$

The first term in this equation corresponds to the relative density in a completely undisturbed lattice, thus it necessarily equals unity. Neglecting terms of order ε^2 and higher this gives

$$\rho_{i\pm 1}^* \approx 1 + VW'(dx)\varepsilon$$

Thus $k = mW'(dx)/\rho_0$. Putting this together with [equation \(A.15\)](#) gives

$$f_x \approx -\frac{V\rho_0 c_0^2 W'(dx)}{\gamma} \left(\frac{(1 + k\varepsilon)^\gamma - 1}{(1 + k\varepsilon)^2} - \frac{(1 - k\varepsilon)^\gamma - 1}{(1 - k\varepsilon)^2} \right) \quad (\text{A.19})$$

using the binomial approximation

$$f_x \approx -\frac{V\rho_0 c_0^2 W'(dx)}{\gamma} \left(\frac{\gamma k\varepsilon}{(1 + k\varepsilon)^2} + \frac{\gamma k\varepsilon}{(1 - k\varepsilon)^2} \right) \quad (\text{A.20})$$

$$\approx -(V\rho_0 c_0^2 W'(dx)k)\varepsilon \underbrace{\left(\frac{1}{(1 + k\varepsilon)^2} + \frac{1}{(1 - k\varepsilon)^2} \right)}_{\approx 2 + 6(k\varepsilon)^2 + O((k\varepsilon)^4)} \quad (\text{A.21})$$

$$\approx -2(V\rho_0 c_0^2 W'(dx)k)\varepsilon \quad (\text{A.22})$$

Thus the force is (at least for small displacements) linear in the displacement ε . We can use $\mathbf{f} = -\nabla_\varepsilon \Phi(\varepsilon)$ to write an equivalent potential which would lead to the force above. It is a simple harmonic potential

$$\Phi(\varepsilon) = \phi \varepsilon^2 \quad (\text{A.23})$$

where $\phi = V\rho_0 c_0^2 W'(dx)k$. The constant ϕ captures how steeply the potential grows as we impose a given displacement, it depends on the parameters of the simulation such as smoothing length, density and speed of sound. Plugging in the value of k we calculated earlier gives

$$\phi = \rho_0 (Vc_0 W'(dx))^2 \quad (\text{A.24})$$

In our simulations we define the average particle spacing $dx = h/s$ where s typically is given a value of 1.3. This means we can write ϕ in terms of either dx or h . Further, the factor $W'(dx)$ can be simplified by noticing we can rewrite the kernel as a product of a self similar

part $\Omega(q = r/h)$ and a normalization term $1/h^d$ giving $W(r, h) = (1/h^d)\Omega(r/h)$.

$$W'(dx) = \frac{dW(r, h)}{dr} \Big|_{r=dx, h=dx s} \quad (\text{A.25})$$

$$= \frac{d}{dr} \left(\frac{1}{h^2} \Omega(q) \right) \Big|_{r=dx, h=dx s} \quad (\text{A.26})$$

$$= \frac{1}{h^d} \frac{d\Omega(q)}{dq} \frac{dq}{dr} \Big|_{r=dx, h=dx s} \quad (\text{A.27})$$

$$= \frac{1}{h^{d+1}} \frac{d\Omega(q)}{dq} \Big|_{q=1/s} \quad (\text{A.28})$$

$$= \frac{1}{h^{d+1}} \Omega' \left(\frac{1}{s} \right) \quad (\text{A.29})$$

where d is the spacial dimension. Thus $\Omega'(1/s)$ is a constant (which depends on the choice of smoothing kernel). Noticing $V = dx^d = (h/s)^d$ allows us to write

$$\phi = \frac{\rho_0 c_0^2}{h^2} \frac{\Omega'(s^{-1})^2}{s^{2d}} \quad (\text{A.30})$$

A.3.1 General Dimensions

Consider the nearest neighbour particles j around a particle i . If the position of i is displaced by some small amount ϵ ($|\epsilon| \ll 1$) what happens to the force felt by i ? Let $\mathbf{x}'_i = \mathbf{x}_i + \epsilon$ be the displaced position. Firstly The densities of all nearby particles will change slightly. The relative density for a nearest neighbour particle is

$$\rho_j^r = V \sum_k W_{kj} \quad (\text{A.31})$$

$$= V \left(W(|\mathbf{x}'_i - \mathbf{x}_j|) + \sum_{k \neq i} W_{kj} \right) \quad (\text{A.32})$$

$$= V \left(W(|\mathbf{x}_i - \mathbf{x}_j|) + \epsilon \cdot \nabla W_{ij} + O(|\epsilon|^2) + \sum_{k \neq i} W_{kj} \right) \quad (\text{A.33})$$

$$= 1 + V \epsilon \cdot \nabla W_{ij} + O(|\epsilon|^2) \quad (\text{A.34})$$

And for the particle i itself we have

$$\rho_i^r = 1 + V \epsilon \cdot \sum_j \nabla W_{ij} \quad (\text{A.35})$$

but since the unperturbed particle positions are at equilibrium $\sum_j \nabla W_{ij}$ is zero. Therefore up to first order $\rho_i^r = 1$. Similarly we get for the number densities that $\sigma_i = 1/V$ and

$$\sigma_j = \frac{1}{V} + \epsilon \cdot \nabla W_{ij}. \quad (\text{A.36})$$

Plugging these densities into the SPH equation for the pressure force gives us

$$-(\nabla P)_i = -\frac{1}{V} \sum_j \left(\frac{P_i}{\sigma_i^2} + \frac{P_j}{\sigma_j^2} \right) \nabla W_{ij}. \quad (\text{A.37})$$

Because the pressure is given by the Tait equation (equation (3.28)) and $\rho_i^r = 1$ we know $P_i = 0$ which gives

$$-(\nabla P)_i = -\frac{1}{V} \sum_j \frac{c_0^2 \rho_0}{\gamma} \frac{(1 + V \epsilon \cdot \nabla W_{ij})^\gamma - 1}{\left(\frac{1}{V_0} + \epsilon \cdot \nabla W_{ij} \right)^2} \nabla W_{ij} \quad (\text{A.38})$$

$$= -V \sum_j \frac{c_0^2 \rho_0}{\gamma} \frac{(1 + V \epsilon \cdot \nabla W_{ij})^\gamma - 1}{(1 + V \epsilon \cdot \nabla W_{ij})^2} \nabla W_{ij} \quad (\text{A.39})$$

$$(\text{A.40})$$

Using the binomial approximation give

$$-(\nabla P)_i = -V \sum_j \frac{c_0^2 \rho_0}{\gamma} \frac{\gamma V \epsilon \cdot \nabla W_{ij}}{(1 + V \epsilon \cdot \nabla W_{ij})^2} \nabla W_{ij} \quad (\text{A.41})$$

whose series expansion, when truncated to first order in ϵ , yields

$$-(\nabla P)_i = -(V c_0)^2 \rho_0 \sum_j \epsilon \cdot \nabla W_{ij} \nabla W_{ij} \quad (\text{A.42})$$

$$= -(V c_0)^2 \rho_0 \sum_j \left(\frac{\partial W_{ij}}{\partial x_{ij}} \right)^2 (\epsilon \cdot \hat{\mathbf{e}}_{ij}) \hat{\mathbf{e}}_{ij}. \quad (\text{A.43})$$

$$(\text{A.44})$$

For certain configurations of the nearest neighbouring particles we get that $\sum_j (\epsilon \cdot \hat{\mathbf{e}}_{ij}) \hat{\mathbf{e}}_{ij} = \lambda \epsilon$. For instance, in 2D a regular hexagonal packing gives $\lambda = 3$ and in 3D a hexagonal close pack gives $\lambda = 4$. This means that the force felt by a displaced particle i up to first order is given by

$$-(\nabla P)_i = -(V c_0 W'(a \Delta x))^2 \rho_0 \lambda \epsilon, \quad (\text{A.45})$$

where a is a factor accounting for the fact that in a hexagonal packing the particles are not quite Δx away from each other. This implies the particles can be thought of as sitting in a quadratic potential where the potential is given by

$$\Phi(\epsilon) = \frac{1}{2} (V c_0 W'(a \Delta x))^2 \rho_0 \lambda \epsilon^2 \quad (\text{A.46})$$

This form is very similar to that obtained in 1D (unsurprisingly), indeed if one checks for one dimension $\lambda = 2, a = 1$ giving us our result for 1D. Since $V = \Delta x^d$ and $\Delta x = h/s$ we get

$$\Phi(\epsilon) = \frac{1}{2} \left(\left(\frac{h}{s} \right)^d c_0 W' \left(\frac{ah}{s} \right) \right)^2 \rho_0 \lambda \epsilon^2 \quad (\text{A.47})$$

Expanding W' into its normalization and dimensionless part we obtain

$$\Phi(\epsilon) = \frac{1}{2} \left(\frac{1}{hs^d} c_0 \Omega' \left(\frac{ah}{s} \right) \right)^2 \rho_0 \lambda \epsilon^2 \quad (\text{A.48})$$

A.4 Derivation of Kernel Gradient - $\nabla\chi_i$

The following is a derivation for including the Boundary Integral Method formulation for wall boundaries into SPH using the corrected Hu & Adams type kernel (see [section 5.4](#)). We begin by deriving the form for the gradient of this corrected kernel since this will be needed to complete the derivation. The Hu & Adams kernel is defined by

$$\chi_i^{(\text{HA})}(\mathbf{x}) = \frac{W_i(\mathbf{x})}{\sigma(\mathbf{x})}, \quad (\text{A.49})$$

where

$$\sigma(\mathbf{x}) = \sum_j W_j(\mathbf{x}). \quad (\text{A.50})$$

However near the edge of the domain (if we do not use ghost particles) σ becomes spuriously lower. To counter this effect we divide σ by the normalization factor γ from [\[37\]](#), giving

$$\chi_i(\mathbf{x}) = \frac{W_i(\mathbf{x})\gamma(\mathbf{x})}{\sigma(\mathbf{x})}. \quad (\text{A.51})$$

It is clear from the definition of χ_i that

$$\sum_i \chi_i(\mathbf{x}) = \frac{\gamma(\mathbf{x})}{\sigma(\mathbf{x})} \sum_i W_i(\mathbf{x}) = \gamma(\mathbf{x}) \quad \forall \mathbf{x}, \quad (\text{A.52})$$

which in the interior of the domain Ω gives $\sum_i \chi_i(\mathbf{x}) \equiv 1$. Next we form a symmetrized form for $\nabla_{\mathbf{x}}\chi_i(\mathbf{x})$ which we will use to form symmetric SPH estimates for gradient quantities.

$$\nabla_{\mathbf{x}}\chi_i(\mathbf{x}) = 1 \cdot \nabla_{\mathbf{x}}\chi_i(\mathbf{x}) - \chi_i(\mathbf{x}) \cdot \nabla_{\mathbf{x}}1, \quad (\text{A.53})$$

from using the chain rule. Next we substitute in $1 = (\sum_i \chi_i(\mathbf{x})) / \gamma(\mathbf{x})$ giving

$$\nabla_{\mathbf{x}}\chi_i(\mathbf{x}) = \frac{\sum_j \chi_j(\mathbf{x})}{\gamma(\mathbf{x})} \cdot \nabla_{\mathbf{x}}\chi_i(\mathbf{x}) - \chi_i(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \left(\frac{\sum_j \chi_j(\mathbf{x})}{\gamma(\mathbf{x})} \right), \quad (\text{A.54})$$

For clarity, unless explicitly stated the dependencies of interest will be on $\mathbf{x}$ and arguments will be dropped so $\chi_i(\mathbf{x})$, $\nabla_{\mathbf{x}}$ and $\sigma(\mathbf{x})$ become χ_i , ∇ and σ respectively.

$$\nabla\chi_i = \sum_j \frac{\chi_j \nabla\chi_i}{\gamma} - \chi_i \left[\frac{1}{\gamma} \nabla \left(\sum_j \chi_j \right) + \sum_j \chi_j \nabla \left(\frac{1}{\gamma} \right) \right], \quad (\text{A.55})$$

where all factors have been brought inside the sum in the first term and the product rule has been used to split the second.

$$\nabla \chi_i = \sum_j \frac{\chi_j \nabla \chi_i}{\gamma} - \sum_j \frac{\chi_i \nabla \chi_j}{\gamma} - \sum_j \chi_i \chi_j \nabla \left(\frac{1}{\gamma} \right) \quad (\text{A.56})$$

$$= \sum_j \frac{\chi_j \nabla \chi_i}{\gamma} - \sum_j \frac{\chi_i \nabla \chi_j}{\gamma} + \sum_j \frac{\chi_i \chi_j}{\gamma^2} \nabla \gamma. \quad (\text{A.57})$$

Combining the first two terms and noticing $\chi_i/\gamma = W_i/\sigma$ gives

$$\nabla \chi_i = \frac{1}{\gamma} \sum_j (\chi_j \nabla \chi_i - \chi_i \nabla \chi_j) + \sum_j \frac{W_i W_j}{\sigma^2} \nabla \gamma, \quad (\text{A.58})$$

bringing a factor of γ/σ out of the first sum

$$\nabla \chi_i = \frac{\gamma}{\gamma \sigma} \sum_j \left(\chi_j \frac{\sigma}{\gamma} \nabla \chi_i - \chi_i \frac{\sigma}{\gamma} \nabla \chi_j \right) + \sum_j \frac{W_i W_j}{\sigma^2} \nabla \gamma, \quad (\text{A.59})$$

then substituting for $\frac{\sigma}{\gamma} \nabla \chi_i = \nabla \left(\frac{\sigma}{\gamma} \chi_i \right) - \chi_i \nabla \left(\frac{\sigma}{\gamma} \right)$

$$\nabla \chi_i = \frac{1}{\sigma} \sum_j \left(\chi_j \left[\nabla \left(\frac{\sigma}{\gamma} \chi_i \right) - \chi_i \nabla \left(\frac{\sigma}{\gamma} \right) \right] - \chi_i \left[\nabla \left(\frac{\sigma}{\gamma} \chi_j \right) - \chi_j \nabla \left(\frac{\sigma}{\gamma} \right) \right] \right) + \sum_j \frac{W_i W_j}{\sigma^2} \nabla \gamma. \quad (\text{A.60})$$

The second term in each of the square brackets cancels and $\frac{\sigma}{\gamma} \chi_i = W_i$ gives

$$\nabla \chi_i = \frac{1}{\sigma} \sum_j (\chi_j \nabla W_i - \chi_i \nabla W_j) + \sum_j \frac{W_i W_j}{\sigma^2} \nabla \gamma. \quad (\text{A.61})$$

The final term can be simplified by using $\sum_j W_j = \sigma$ so finally

$$\nabla \chi_i = \frac{1}{\sigma} \sum_j (\chi_j \nabla W_i - \chi_i \nabla W_j) + \frac{W_i}{\sigma} \nabla \gamma. \quad (\text{A.62})$$

Comparing this to the Hu & Adams kernel we see

$$\nabla \chi_i = \nabla \chi_i^{(\text{HA})} + \frac{W_i}{\sigma} \nabla \gamma. \quad (\text{A.63})$$

A.5 Derivation of SPH Gradient Estimate - $\langle \nabla A \rangle$

We now use [equation \(A.62\)](#) in formulating the SPH estimate of a gradient quantity. The SPH estimate for the gradient of a quantity $A(\mathbf{x})$ is given by

$$\langle \nabla_{\mathbf{x}} A \rangle_i = \frac{1}{V_i \gamma_i} \int_{\Omega \cap \Omega_i} \chi_i(\mathbf{y}) \nabla_{\mathbf{y}} A(\mathbf{y}) \, d\mathbf{y}, \quad (\text{A.64})$$

which can be split using integration by parts to give

$$\langle \nabla A \rangle_i = \frac{1}{V_i \gamma_i} \int_{\Omega \cap \Omega_i} \nabla (\chi_i A) - A \nabla \chi_i \, d\mathbf{y}, \quad (\text{A.65})$$

Again, where the arguments $\mathbf{y}$, $\mathbf{x}$ are clear from the context they are dropped to improve clarity.

$$\langle \nabla A \rangle_i = \frac{1}{V_i \gamma_i} \int_{\Omega \cap \Omega_i} \nabla (\chi_i A) \, d\mathbf{y} - \frac{1}{V_i \gamma_i} \int_{\Omega \cap \Omega_i} A \nabla \chi_i \, d\mathbf{y}, \quad (\text{A.66})$$

$$= \underbrace{\frac{1}{V_i \gamma_i} \int_{\partial(\Omega \cap \Omega_i)} \chi_i A \, \mathbf{n} dS}_{:=S_i} - \underbrace{\frac{1}{V_i \gamma_i} \int_{\Omega \cap \Omega_i} A \nabla \chi_i \, d\mathbf{y}}_{:=U_i}. \quad (\text{A.67})$$

The first integral S_i has been converted to a surface integral using the divergence theorem (actually its scalar equivalent). If the region of integration is in the interior of Ω , i.e. $\Omega \cap \Omega_i = \Omega_i$ then this integral is necessarily zero because everywhere on the boundary $\chi_i = 0$. However, if the particle i is near $\partial\Omega$ then on the part of $\partial\Omega$ which lies inside Ω_i , namely $\partial\Omega \cap \Omega_i$, the integrand is not zero and we cannot discount this term.

Focusing first on the volume integral term U_i we substitute in [equation \(A.62\)](#) giving

$$U_i = \frac{1}{V_i \gamma_i} \int A \left(\frac{1}{\sigma} \sum_j (\chi_j \nabla W_i - \chi_i \nabla W_j) + \frac{W_i}{\sigma} \nabla \gamma \right) d\mathbf{y} \quad (\text{A.68})$$

$$= \frac{1}{V_i \gamma_i} \int A \frac{1}{\sigma} \sum_j (\chi_j \nabla W_i - \chi_i \nabla W_j) \, d\mathbf{y} + \frac{1}{V_i \gamma_i} \int A \frac{W_i}{\sigma} \nabla \gamma \, d\mathbf{y} \quad (\text{A.69})$$

$$= \frac{1}{V_i \gamma_i} \sum_j \int A \frac{1}{\sigma} (\chi_j \nabla W_i - \chi_i \nabla W_j) \, d\mathbf{y} + \frac{1}{V_i \gamma_i} \int A \frac{W_i}{\sigma} \nabla \gamma \, d\mathbf{y} \quad (\text{A.70})$$

Now we approximate the terms in the first integral. Since W_j is highly peaked around $\mathbf{x}_j$ most of the weight in the integral comes from values of γ , σ and A which are approximately equal to γ_j , etc. Combined with the fact $\int W_j = \gamma_j$ this gives

$$\int_{\Omega \cap \Omega_i} \frac{A(\mathbf{y}) \gamma(\mathbf{y})}{\sigma(\mathbf{y})^2} W_j(\mathbf{y}) \nabla_{\mathbf{y}} W_i(\mathbf{y}) \, d\mathbf{y} \approx \frac{A_j \gamma_j^2}{\sigma_j^2} \nabla W_{ij},$$

and similarly for the second term. After using $\nabla W_{ij} = -\nabla W_{ji}$ to pull the gradients out of the brackets this gives

$$U_i \approx \frac{1}{V_i \gamma_i} \sum_j \left(\frac{A_i \gamma_i^2}{\sigma_i^2} + \frac{A_j \gamma_j^2}{\sigma_j^2} \right) \nabla W_{ij} + \frac{1}{V_i \gamma_i} \int A \frac{W_i}{\sigma} \nabla \gamma \, d\mathbf{y}, \quad (\text{A.71})$$

The second integral can also be approximated in exactly the same way as the first two so we have

$$U_i \approx \frac{1}{V_i \gamma_i} \sum_j \left(\frac{A_i \gamma_i^2}{\sigma_i^2} + \frac{A_j \gamma_j^2}{\sigma_j^2} \right) \nabla W_{ij} + \frac{1}{V_i \gamma_i} \frac{A_i \gamma_i}{\sigma_i} (\nabla \gamma)_i. \quad (\text{A.72})$$

Next it remains to derive an approximation for the surface integral term S_i

$$S_i = \frac{1}{V_i \gamma_i} \int A \chi_i \mathbf{n} dS \quad (\text{A.73})$$

$$= \frac{1}{V_i \gamma_i} \int A \frac{W_i \gamma}{\sigma} \mathbf{n} dS \quad (\text{A.74})$$

$$(\text{A.75})$$

by placing boundary particles along the entire boundary $\partial\Omega$ then we can approximate this integral with a sum of boundary particles $b \in B$ so that

$$S_i \approx \frac{1}{V_i \gamma_i} \sum_b \frac{A_b \gamma_b}{\sigma_b} W_{ib} \mathbf{n}_b dS \quad (\text{A.76})$$

where dS is the area (length in 2D) that boundary particle represents. Thus giving the approximation for the gradient of $A(\mathbf{x}_i)$ as

$$\nabla A_i = \frac{1}{V_i \gamma_i} \sum_j \left(\frac{A_i \gamma_i^2}{\sigma_i^2} + \frac{A_j \gamma_j^2}{\sigma_j^2} \right) \nabla W_{ij} + \frac{1}{V_i \gamma_i} \frac{A_i \gamma_i}{\sigma_i} \nabla \gamma_i + \frac{1}{V_i} \sum_b \frac{A_b \gamma_b}{\sigma_b} W_{ib} \mathbf{n}_b dS. \quad (\text{A.77})$$

In (A.77) we need to know both γ and its gradient. Since

$$(\nabla \gamma)_i = \int_{\Omega \cap \Omega_i} \nabla W_i d\mathbf{y} = \int_{\partial\Omega \cap \Omega_i} W_i \mathbf{n} dS$$

we can use the boundary particles B to estimate $(\nabla \gamma)_i$ as

$$(\nabla \gamma)_i = \sum_b W_{ib} \mathbf{n}_b dS \quad (\text{A.78})$$

which when substituted into (A.77) and simplified gives us

$$\nabla A_i = \frac{1}{V_i \gamma_i} \sum_j \left(\frac{A_i \gamma_i^2}{\sigma_i^2} + \frac{A_j \gamma_j^2}{\sigma_j^2} \right) \nabla W_{ij} + \frac{1}{V_i \gamma_i} \sum_b \left(\frac{A_i \gamma_i}{\sigma_i} + \frac{A_b \gamma_b}{\sigma_b} \right) W_{ib} \mathbf{n}_b dS. \quad (\text{A.79})$$

The sum over boundary particles represents the correction to the gradient due to being near a boundary. Far ($> 2h$) from the boundary $\gamma_i = 1$ and $W_{ib} = 0 \forall b$ thus we recover the standard Hu & Adams gradient estimate.

A.6 Derivation of Superficial Velocity Time Dependence - $v_s(t)$

We expect the superficial velocity $u_s = \langle u \rangle$ as a function of time to asymptote to a constant value, i.e.

$$\lim_{t \rightarrow \infty} u_s = \text{const.}$$

this is because, although the system is being driven by gravity the viscous forces experienced by the fluid acts to damp the motion and eventually will cancel the acceleration of gravity.

Assuming low Reynolds number flow we can write

$$F = \rho V_f g - K u_s, \quad (\text{A.80})$$

which represents the forces acting on the fluid (considered as one body), first is simply acceleration due to gravity and the second term is Stokes drag. This gives us a differential equation

$$\frac{du_s}{dt} = g - \frac{K}{\rho V_f} u_s. \quad (\text{A.81})$$

This can be easily solved. First, separate the variables

$$\int \frac{du_s}{g - \frac{K}{\rho V_f} u_s} = \int dt, \quad (\text{A.82})$$

$$(\text{A.83})$$

Then setting $K' = \frac{K}{\rho V_f}$ and $u' = g - K' u_s$ implies

$$\begin{aligned} -\frac{1}{K'} \int \frac{du'}{u'} &= t + C, \\ \Rightarrow \log(u') &= -K'(t + C) \\ \Rightarrow g - K' u_s &= e^{-K'(t+C)} \\ \Rightarrow u_s &= \frac{1}{K'} \left(g - e^{-K'(t+C)} \right) \\ \Rightarrow u_s &= \frac{\rho V_f g}{K} \left(g - e^{-K't} e^{-K'C} \right) \end{aligned} \quad (\text{A.84})$$

where C is introduced as a constant of integration. Using the initial condition $u_s(0) = 0$ we get

$$0 = \frac{\rho V_f g}{K} \left(g - e^{-K'C} \right) \quad (\text{A.85})$$

$$\Rightarrow g - e^{-K'C} = 0 \quad (\text{A.86})$$

$$\Rightarrow e^{-K'C} = g \quad (\text{A.87})$$

Substituting A.87 into A.84 gives

$$u_s = \frac{\rho V_f g}{K} \left(1 - \exp \left(-\frac{K}{\rho V_f} t \right) \right). \quad (\text{A.88})$$

V_f , the volume of fluid in the system clearly depends on the saturation θ_s and porosity ϵ as $V_f = V \epsilon \theta_s$ where V is the total volume of the system. Also, the drag coefficient we might expect to depend on the dynamic viscosity μ , for dimensional reasons then $K = \mu L C_d$ where L is a characteristic length scale and C_d is the dimensionless drag coefficient. Substituting these in gives us

$$u_s = \frac{\rho V \epsilon \theta_s g}{\mu L C_d} \left(1 - \exp \left(-\frac{\mu L C_d}{\rho V \epsilon \theta_s} t \right) \right). \quad (\text{A.89})$$

Which has the asymptotic value

$$\lim_{t \rightarrow \infty} u_s = \frac{\rho V \epsilon \theta_s g}{\mu L C_d}. \quad (\text{A.90})$$

One should be concerned that this apparently depends on the total volume of the system. We wouldn't expect doubling the system to double the final velocity if the system is homogenous. The key point to note here is that there is a volume dependence hidden in C_d - the drag coefficient - meaning C_d is actually $C_d(V)$. What form does this take? It is simple to reason that

$$C_d(V) = \tilde{c}_d \cdot V \quad (\text{A.91})$$

where $\tilde{c}_d > 0$ is a constant. To show that C_d must be linear and distributive note that since we do not expect the terminal velocity to depend explicitly or implicitly on the system size we have

$$u_{s,V_1}(\infty) = u_{s,V_2}(\infty) \Rightarrow \frac{\rho V_1 \epsilon \theta_s g}{\mu L C_d(V_1)} = \frac{\rho V_2 \epsilon \theta_s g}{\mu L C_d(V_2)}. \quad (\text{A.92})$$

which simplifies to

$$\frac{V_1}{V_2} = \frac{C_d(V_1)}{C_d(V_2)}, \quad (\text{A.93})$$

writing $V_1 = \lambda V_2$ and multiplying both sides by $C_d(V_2)$ implies

$$\lambda C_d(V_2) = C_d(\lambda V_2), \quad (\text{A.94})$$

thus C_d is linear and must be of the form [equation \(A.91\)](#). The linearity of C_d implies it is also distributive which makes sense since a system twice as large, for example, should have twice as much actual drag. From this it is clear that $\tilde{c}_d$ represents the *drag coefficient per unit volume*. There is nothing to stop $\tilde{c}_d$ being a function of the porosity of the system (amongst other things). So finally we have

$$u_s = \frac{\rho \epsilon \theta_s g}{\mu L \tilde{c}_d} \left(1 - \exp \left(-\frac{\mu L \tilde{c}_d}{\rho \epsilon \theta_s} t \right) \right). \quad (\text{A.95})$$

A.7 Derivation of Dual Population Dispersion - $D(\tau)$

Assume that the flow consists of two distinct 'populations' a static and a flowing portion. Let $g(\Delta x)$ be the probability density function (pdf) of the (axial) displacement for a particle in the static phase after time τ . Similarly let $f(\Delta x)$ be the pdf of the (axial) displacement for a particle in the flowing phase after time τ . We know, since it is static, that $g(\Delta x) = \delta(\Delta x)$. Let the flowing phase move with a seepage velocity v'_s . This implies that the mean of f is $\mu_f = v'_s \tau$. Also let the variance of f be given by $\sigma_f^2 = C\tau(\alpha + 1)$ where $-1 \leq \alpha$ (this neglects the effect of velocity autocorrelation at short times).

The overall pdf of displacements is given by $h(\Delta x) = pg(\Delta x) + (1 - p)f(\Delta x)$. We can then calculate the moments of $h(\Delta x)$.

$$\mu_h = \int_{\mathbb{R}} (pg(x) + (1-p)f(x))x \, dx \quad (\text{A.96})$$

$$= p \int_{\mathbb{R}} g(x)x \, dx + (1-p) \int_{\mathbb{R}} f(x)x \, dx \quad (\text{A.97})$$

$$= p\mu_g + (1-p)\mu_f. \quad (\text{A.98})$$

Next the variance of h is given by

$$\sigma_h^2 = \int_{\mathbb{R}} (pg(x) + (1-p)f(x))x^2 \, dx - \mu_h^2, \quad (\text{A.99})$$

$$= p \int_{\mathbb{R}} g(x)x^2 \, dx + (1-p) \int_{\mathbb{R}} f(x)x^2 \, dx - \mu_h^2, \quad (\text{A.100})$$

$$= p(\sigma_g^2 - \mu_g^2) + (1-p)(\sigma_f^2 - \mu_f^2) - (p^2\mu_g^2 + (1-p)^2\mu_f^2 + 2p(1-p)\mu_g\mu_f), \quad (\text{A.101})$$

which, by substituting $\sigma_g^2 = \mu_g = 0$, $\sigma_f^2 = C\tau^\alpha$ and $\mu_f = v'_s\tau$ then simplifying yields finally

$$\sigma_h^2 = C(1-p)\tau^{(\alpha+1)} + p(1-p)(v'_s)^2\tau^2 \quad (\text{A.102})$$

thus

$$D = C(1-p)\tau^\alpha + p(1-p)(v'_s)^2\tau. \quad (\text{A.103})$$

A.8 Derivation of the Green-Kubo Relation

The Green-Kubo relation states that

$$D = \int_0^\infty \phi(t) \, dt, \quad (\text{A.104})$$

where $\phi(t)$ is the velocity autocorrelation

$$\phi(t) = \langle u(t)u(t+\tau) \rangle_\tau. \quad (\text{A.105})$$

Physically the Green-Kubo relation is a consequence of the fluctuation-dissipation theorem. Although the derivation is published in [62] it is presented here for completeness. Mathematically it can be shown by using the property of convolutions that

$$\int_{\mathbb{R}} (f * g)(x) \, dx = \left(\int_{\mathbb{R}} f(x) \, dx \right) \left(\int_{\mathbb{R}} g(x) \, dx \right), \quad (\text{A.106})$$

which follows from Fubini's theorem regarding the reordering of iterated integrals and the translation invariance of an integral over $\mathbb{R}$.

Begin by writing

$$D = \lim_{t \rightarrow \infty} \frac{1}{2t} \langle \zeta_0(t)^2 \rangle, \quad (\text{A.107})$$

which is the usual definition of the dispersion (or diffusion) coefficient. Since ζ is given by the

integral of the velocity we can write

$$D = \lim_{t \rightarrow \infty} \frac{1}{2t} \left\langle \left(\int_0^t u(\tau) d\tau \right) \left(\int_0^t u(\tau) d\tau \right) \right\rangle. \quad (\text{A.108})$$

Using [equation \(A.106\)](#) this can be transformed to

$$D = \lim_{t \rightarrow \infty} \frac{1}{2t} \left\langle \int_0^t (u * u)(\tau) d\tau \right\rangle, \quad (\text{A.109})$$

$$= \lim_{t \rightarrow \infty} \frac{1}{2t} \int_0^t \langle (u * u)(\tau) \rangle d\tau, \quad (\text{A.110})$$

$$= \int_0^\infty \phi(\tau) d\tau. \quad \square \quad (\text{A.111})$$

A.9 Derivation of Striated Geometry Characteristic Length

For a sample of measurements $\{X_i\}$ of a random variable X the geometric mean is given by

$$\langle X \rangle_{\text{geom}} = \sqrt[N]{\prod_{i=1}^N X_i}.$$

Taking logarithms of each side this becomes

$$\log(\langle X \rangle_{\text{geom}}) = \frac{1}{N} \sum_{i=1}^N \log(X_i).$$

For a real random variable $a \leq X \leq b$ with a probability density function $f(x)$ we can write this as

$$\log(\langle X \rangle_{\text{geom}}) = \int_a^b f(x) \log(x) dx. \quad (\text{A.112})$$

For our particles being placed in the stratified geometry we have $f(R) \propto 1/R$ because for a larger radius we can fit fewer particles into the x period. This is easily normalized by calculating

$$\int_a^b \frac{A}{r} dr = 1,$$

to give

$$A = \frac{1}{\log(b) - \log(a)}.$$

This $f(R)$ can then be substituted into [equation \(A.112\)](#) giving

$$\log(\langle R \rangle_{\text{geom}}) = A \int_a^b \frac{\log(r)}{r} dr. \quad (\text{A.113})$$

This can be expanded using integration by parts to give

$$\log(\langle R \rangle_{\text{geom}}) = A(\log^2(b) - \log^2(a)) - A \int_a^b \frac{\log(r)}{r} dr, \quad (\text{A.114})$$

but the last term is just the original integral so we get finally

$$\log \langle R \rangle_{\text{geom}} = \frac{1}{2} \frac{\log^2(b) - \log^2(a)}{\log(b) - \log(a)} \quad (\text{A.115})$$

$$\Rightarrow \langle R \rangle_{\text{geom}} = \exp \left(\frac{1}{2} \frac{\log^2(b) - \log^2(a)}{\log(b) - \log(a)} \right). \quad (\text{A.116})$$

For the given values of $a = R_{\min} = 0.25$ cm and $b = R_{\max} = 2$ cm this gives a length scale of 0.7071 cm.

Appendix B

Miscellaneous

B.1 Software

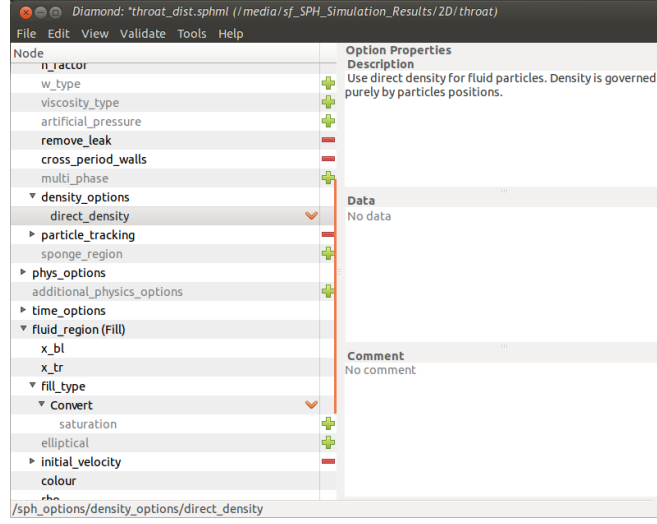


Figure B.1: The diamond GUI [46] provides a user friendly way to set simulation parameters

The SPH simulation software, surface processing software and post-processing software were all written ‘in house’ in C++. The SPH code uses the following libraries; MPICH2, PETSc, MuParser, libspud [46]. The analysis scripts, including *a posteriori* simulation of molecular diffusion, were written using the Python programming language (predominantly version 2.7.0, some 3.3.0).

The various plots and graphs in this thesis were produced using the Matplotlib library (versions 1.2.0 and 1.3.1) for Python [50]. The diagrams were drawn using the TikZ/pgf package (versions 2.1.0 & 3.0.0) for the L^AT_EX typesetting environment, and 3D visualizations were produced using the VTK library (version 5.6) & ParaView software (version 3.98.1).

The libspud library allows the use of the Diamond GUI for setting simulation parameters in a user friendly fashion, an example screenshot is shown in figure B.1. Additionally the included utility ‘spud-set’ allows one to programmatically set options, allowing simulations with varying parameters to be entered quickly using a Python script.

B.2 Publications

Some of the work contained in this thesis has been presented at conferences, published, or submitted for publication. The following is a list of articles which have, so far, been published

based on work from this thesis

- SPH Simulation of Packed-beds and Columns Applied to Heap-leaching. *D.J. Barker, G. Parameswaran and S.J. Neethling*. 9th International Conference on CFD in the Minerals and Process Industries, *Ed: C. Solnordal et al*, July 2012. [Peer reviewed] [6]
- Application of B-Splines and Curved Geometries to Boundaries in SPH. *D.J. Barker, P.R. Brito-Parada and S.J. Neethling*. International Journal for Numerical Methods in Fluids. Vol. 76, Issue 1, pages 51 - 68. [4]
- Dispersion of Unsaturated Unsteady Flow in 2D Periodic Packings. *D.J. Barker and S.J. Neethling*. MEi Computational Modelling '13, *Ed: B. Wills*, June 2013. [5]