University of Strathclyde
Department of Mechanical Engineering

Molecular Dynamics Simulation in Arbitrary Geometries for Nanoscale Fluid Mechanics

Graham Bruce Macpherson

A thesis presented in fulfilment of the requirements
for the degree of Doctor of Philosophy

2008

Declaration of author's rights

Acknowledgements

Since giving the talk about MEMS and rarefied gas dynamics that sparked my interest in this field of research, Jason Reese has been a constant influence. I would like to thank him for being an inspiring and supportive supervisor and for accepting my assurances that I would, eventually, produce some papers and results.

I am grateful to the Miller Foundation and the University of Strathclyde for their generosity in providing the scholarship that has funded me during my doctorate.

A large part of the success of my research has been due to my use of OpenFOAM as a simulation code. I am indebted to Chris Greenshields of OpenCFD Ltd. and the University of Strathclyde and Andy Heather, Mattijs Janssens and Henry Weller of OpenCFD Ltd. for creating this excellent software and their help with its use. Dave Emerson, Rob Barber and Bill Smith of the STFC Daresbury Laboratory have provided me with help and advice from the outset. My thanks also go to Tom Scanlon as my second supervisor and for teaching me so much about thermodynamics and fluid mechanics as an undergraduate.

Abstract

Simulations of nanoscale systems where fluid mechanics plays an important role are required to help design and understand nano-devices and biological systems. A simulation method which hybridises molecular dynamics (MD) and continuum computational fluid dynamics (CFD) is demonstrated to be able to accurately represent the relevant physical phenomena and be computationally tractable.

An MD code has been written to perform MD simulations in systems where the geometry is described by a mesh of unstructured arbitrary polyhedral cells that have been spatially decomposed into irregular portions for parallel processing. The MD code that has been developed may be used for simulations on its own, or may serve as the MD component of a hybrid method. The code has been implemented using OpenFOAM, an open source C++ CFD toolbox (www.openfoam.org).

Two key enabling components are described in detail. 1) Parallel generation of initial configurations of molecules in arbitrary geometries. 2) Calculation of intermolecular pair forces, including between molecules that lie on mesh portions assigned to different, and possibly non-neighbouring processors.

To calculate intermolecular forces, the spatial relationship of mesh cells is calculated once at the start of the simulation and only the molecules contained in cells that have part of their surface closer than a cut-off distance are required to interact. Interprocessor force calculations are carried out by creating local copies of molecules from other processors in a layer around the processor in question. The process of creating these copied molecules is described in detail.

A case study of flow in a realistic nanoscale mixing channel, where the geometry is drawn and meshed using engineering CAD tools, is simulated to demonstrate the capabilities of the code for complex simulations.

Preface

Original contribution

Chapters 6, 7, 11 and 12 and appendices C, D and E, represent the main original technical component of the thesis and constitute its unique contribution to knowledge. The contents of these chapters has been accepted to a peer-reviewed journal, references [1] and [2]. All of the work presented is solely my own, except where otherwise attributed by citations in the text, with two exceptions:

- the parallelisation of the initial molecular configuration utility in chapter 6 and the development of the optimised anchor position in section 6.4 was carried out in conjunction with Matthew K. Borg of the University of Strathclyde;
- the particle tracking algorithm described in section 8.3 was developed by Niklas Nordin (Scania CV AB, Sweden, formerly of Chalmers University of Technology, Sweden) and Henry Weller (OpenCFD Ltd, UK). This section is adapted from reference [3]; the text and images of this paper were produced by me. The algorithm was previously unpublished and my involvement with this paper arose from a need to formally document it. This was required to convince referees of a draft of reference [2] that the cost and complexity of determining which cell a molecule occupied did not make the proposed intermolecular force calculation algorithm impractically expensive.

Inclusions to provide context

The work presented in this thesis is the subject of ongoing research and development. Some sections have been included that describe work that has not been implemented and tested. They are intended to place what has been done in a wider context and provide a more complete and coherent picture of what future development is required and some of the goals that motivate it. For example, generalised open boundaries and driving volumes as described in sections 9.3 to 9.5, have not been implemented, but are necessary future steps.

Some sections are explained in detail because, although they are not necessarily original, there is no literature source for them. They will be useful to anyone using, modifying or extending this work. These sections are the derivation of reduced units (appendix A), the derivation of the force equation for the shifted force Lennard-Jones potential in general reduced unit form (appendix B) and

the explanation of the cause of energy cascades when too large an integration timestep is used (section 8.2.1).

Appendix F (adapted from [4]) does not fit into the main body of the thesis because it deals with gas dynamics and the results were obtained using a precursor MD code to the one presented in the thesis. This appendix is relevant, however, in the context of fluid dynamics simulation by molecular methods, and studies issues applicable to the work of the research group I was based in.

OpenFOAM and programming

Features of the C++ programming language make writing, modifying and maintaining a large and complex piece of software significantly easier. OpenFOAM [5] has been used to implement the simulation algorithms presented in this thesis; it makes extensive use of these features. See appendix G for an introduction to OpenFOAM.

It is difficult to describe many aspects of a software based project in a thesis. The details of the code are too complex to describe meaningfully without listing the code itself. In this case it would be even more difficult to describe because a substantial knowledge of OpenFOAM would be required to provide the context to appreciate and understand the code fully. Therefore, this thesis contains the specification and requirements of the code, the algorithms that have been implemented, and some results that the code has produced. This does, however, obscure the fact that writing 'good' code (well designed and structured, readable, general and reusable, easy to modify and maintain, flexible, modular and extensible) is crucial to success and contains much of the research and engineering skill that has been developed and used.

Note on reading

Readers unfamiliar with the concept of molecular dynamics as a simulation technique should read section 5.1 before reading chapters 2, 4 or beyond.

Sections 13.3 and D.4 comprise sequences of images. These are best viewed page-by-page as a pdf file (see attached CD) to appreciate the transitions.

Graham Macpherson
May 2008

Contents

Contents vi

List of Figures xi

List of Tables xv

I Motivation and requirements 1

1 Introduction 2

1.1 Research objectives 2
1.2 What is the 'nanoscale' 3
1.3 Nano-scale applications 4
1.4 Physical phenomena 5
1.4.1 Surface effects 6
1.4.2 Thermal fluctuations and diffusion 6
1.4.3 Invalidation of continuum equations 6
1.4.4 Electrokinetic effects 7
1.5 Thesis structure 8

2 Experimental data literature 9

2.1 Nano-channels and nano-devices 9
2.1.1 Nanotube and nanopipe flows 9
2.1.2 Electrokinetic devices and flows 10
2.2 Surface effects 10
2.2.1 Surface force experiment 11
2.2.2 Boundary slip 11
2.2.3 Nanobubbles 12

2.2.4 Density depletion 12
2.3 Summary 13

3 Evaluation of simulation techniques 14
3.1 Specification and requirements 14
3.2 Continuum computational fluid dynamics 15
3.3 Molecular dynamics 16
3.4 Lattice Boltzmann 16
3.5 Dissipative particle dynamics 18
3.6 Evaluation and comparison 18

4 Hybrid molecular/continuum simulation 20
4.1 Breakdown of continuum equations 22
4.1.1 Continuum failure 23
4.1.2 Deviation from hydrodynamics and linear response 23
4.2 Coupling MD and continuum regions 24
4.3 Validation 24
4.4 Hybrid simulations in arbitrary geometries 25

II Molecular dynamics in arbitrary geometries 27

5 MD overview and requirements 28
5.1 Molecular dynamics overview 28
5.1.1 Reduced units 29
5.1.2 Intermolecular potentials 30
5.1.3 The Lennard-Jones potential 32
5.1.4 Periodic boundaries 33
5.2 Limitations of conventional molecular dynamics 33
5.2.1 System geometry 33
5.2.2 Neighbour lists 35
5.3 Requirements for arbitrary geometry simulation 37
5.3.1 Parallelisation 38
5.4 Implementation in OpenFOAM 38
5.4.1 Mesh description 39
5.4.2 Continuum solver integration 39
5.4.3 Reduced unit implementation 39

6 Initial molecular configuration 41
6.1 Creating an expanding cube of unit cells . . . 42
6.2 Zone specification: molConfigDict . . . 46
6.3 Generating molecule positions from a unit cell . . . 46
6.4 Optimising the anchor position . . . 50
6.5 A three zone test case . . . 52
6.6 Molecular velocity generation . . . 54
6.7 Avoiding high energy overlaps at zone boundaries . . . 57

7 Intermolecular Force Calculation 60
7.1 Interacting cell identification . . . 60
7.2 Replicated molecule periodicity and parallelisation . . . 62
7.3 Referred molecules and cells . . . 64
7.3.1 Referred molecule . . . 64
7.3.2 Referred cell . . . 64
7.4 Interacting cell identification methods . . . 65
7.5 Intermolecular force calculation procedure . . . 73

8 Integration of equations of motion 75
8.1 Integration algorithms . . . 75
8.1.1 Leapfrog . . . 75
8.2 Choosing an integration timestep . . . 77
8.2.1 Stability: preventing energy cascades . . . 77
8.3 Tracking molecules in arbitrary unstructured meshes . . . 79
8.3.1 Basic particle tracking algorithm . . . 79
8.3.2 Modified tracking algorithm . . . 83
8.3.3 Boundary interactions and parallelisation . . . 87

9 Boundary conditions and driven flows 89
9.1 Simple solid walls . . . 89
9.2 Boundary force walls . . . 90
9.3 Generalised open boundaries . . . 91
9.3.1 Variable permeability periodic boundaries . . . 93
9.4 Thermostats, constraints and driven flows . . . 93
9.5 Setting and feedback control of parameters . . . 95

10 Spatially resolved flow properties **97**
10.1 Spatial and temporal averaging in cells . . . 97
10.2 Measureable properties . . . 99
10.2.1 Gas collision properties . . . 100
10.3 Velocity . . . 100
10.4 Kinetic temperature . . . 101
10.4.1 Other definitions of temperature . . . 104
10.5 Density and mass/mole fraction . . . 104
10.6 Velocity distribution . . . 105
10.6.1 Distribution class . . . 106

III Performance and testing **108**

11 Molecular configuration performance **109**
11.1 Aspect ratio . . . 110
11.2 Parallel performance. . . . 114
11.3 Choosing mesh dimensions and anchor positions . . . 114

12 Force calculation performance **116**
12.1 Accuracy: average pair potential energy . . . 116
12.1.1 Parallel accuracy . . . 118
12.1.2 Distribution of errors . . . 119
12.2 Computational speed . . . 121
12.2.1 Speed comparison with conventional MD code . . . 123
12.3 Discussion . . . 126
12.3.1 Choosing a mesh . . . 126

13 Case study: CAD-derived mixing channel **128**
13.1 Geometry definition . . . 128
13.2 MD preprocessing and simulation . . . 131
13.3 Results . . . 142
13.4 Discussion . . . 170

14 Conclusions and further work **172**
14.1 Conclusions . . . 172
14.2 Future developments . . . 173

14.2.1 Expand range of materials 174
14.2.2 Measurements 174
14.2.3 Boundaries and driven flows 175
14.2.4 Performance improvements 175

Bibliography **177**

IV Appendices **192**

A Reduced unit derivation **193**

B Derivation of shifted force equation **197**

C Neighbour list lifetime prediction **201**

D Build cell interactions **205**
D.1 Building DILs 205
D.2 Creating referred cells 206
D.2.1 Creating patch segments 206
D.2.2 Cell referring iterations 207
D.3 Determining real cells in range of referred cells 208
D.4 Example construction of referred cells 208

E Cell referring transformations **228**

F Non-equilibrium gas dynamics **232**

G Introduction to OpenFOAM **237**
G.1 Overview 237
G.2 Modular code structure 238
G.3 Why C++? 239

List of Figures

4.1 Schematic of an application of a hybrid MD/continuum simulation: complex molecules being transported into a nano channel. 21

5.1 $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of molecules i and j, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $\mathbf{f}_{ij}$ is the force acting on i caused by j. 29
5.2 An overview of the molecular dynamics calculation process. . . . 29
5.3 The Lennard-Jones (LJ) and shifted force LJ intermolecular potentials for argon. 32
5.4 Periodic boundaries: notional and as-implemented. 34

6.1 Filling a single mesh zone comprising two disconnected volumes. 42
6.2 Exploded view of adding a layer of unit cells to a cube. 43
6.3 Constructing the sides of an expanding square. 45
6.4 Placing the optimised anchor at the unit cell position closest to the centre of the volumes in the zone. 51
6.5 Three zone test case mesh and zones: a channel with a constriction. 52
6.6 Three zone test case configuration of molecules. 53
6.7 3D view of the molecules created in the three zone test case. . . 54
6.8 The mesh for the three zone test case decomposed into 4 portions for parallel processing. 55
6.9 The disconnected volumes of the bottom wall and liquid zones residing on processor 1. 55
6.10 Details of removing high energy overlaps at zone boundaries. . . 58

7.1 Interacting cell identification example. 61
7.2 Spatial decomposition for parallel processing: replicated molecules. 62
7.3 Periodic boundaries implemented using replicated molecules . . 63
7.4 An example of a non-parallel, separated boundary. 63

7.5 Errors caused by cut-off radius searching between vertices only. . 66
7.6 Two tetrahedral cells with edges within r_{cut} of each other which searching from vertices cannot establish. 67
7.7 The three possibilities for the closest point on a face to an arbitrary point; the point can be closest to a vertex, an edge, or the face itself. 68
7.8 Projecting the point to be evaluated onto the plane defined by the face centre and normal unit vector. 68
7.9 Defining a coordinate system local to the plane of the face to calculate intersections with face edges. 71
7.10 The closest distance between two edges. 72

8.1 The approach of two molecules with different integration timesteps. 78
8.2 A particle moving from position **a** to **b**, crossing two cell faces. . 80
8.3 The physical interpretation of the value of λ_a. 81
8.4 A mesh with a twisted face which differs from its effective plane 82
8.5 An analogous situation to a particle tracking across a face close to a vertex, in the vicinity of a non-planar face. 83
8.6 A particle moving in concave cell becoming stuck in an infinite loop. 86
8.7 A particle tracking across a face that is translating and rotating. 86

9.1 An example mass addition counter for a face. 93
9.2 A simple flow system employing driving volumes and wall molecules tethered into a crystal 94
9.3 A fluid inlet or outlet created using feedback control. 95

11.1 Tetrahedral test mesh. 110
11.2 The computational time required to generate a given number of molecules with a varying number of cells in the mesh. 111
11.3 The computational time required to fill a mesh of a given number of cells with a varying number of molecules. 111
11.4 Two examples of the mesh used to examine aspect ratio sensitivity. 113
11.5 The computational cost of filling varying aspect ratio cuboids. . 113

12.1 Relative error between $\langle PE \rangle$ for each tetrahedral mesh and $\langle PE \rangle_{gF}$ with different cell interaction build methods. 117

12.2 Relative error in $\langle PE \rangle$ for each tetrahedral mesh simulated in parallel using 12 and 32 processors. 119
12.3 Relative error magnitude distribution. 120
12.4 Absolute error magnitude distribution. 121
12.5 Time taken to build interaction lists at the start of the simulation. 123
12.6 Time taken to calculate intermolecular pair forces between all molecules. 124

13.1 The geometry that the case study is based on. 129
13.2 The Pro/ENGINEER® defining sketch of the mixing channel case study. 133
13.3 Overall geometry dimensions. 134
13.4 3D view of the volume created in Pro/ENGINEER® 135
13.5 The geometry imported into GAMBIT® 135
13.6 New faces are created to define fluid volume inlet and exit faces. 136
13.7 Defining an internal volume for the fluid. 136
13.8 The fluid volume is split into four parts. 137
13.9 Molecule reservoirs are added to the fluid inlets. 137
13.10 The liquid zones of the mesh. 138
13.11 The wall zone of the mesh. 138
13.12 The internal space forming the liquid channel. 139
13.13 The molecules and mesh contained on two processor portions. . 139
13.14 Entries in the molConfigDict dictionary for the case study. . . . 140
13.15 Entries in the potentials dictionary for the case study. 141
13.16 Cuts through the x and z normal planes of the domain. 143
13.17 Comparing the cell and interpolated field data. 143
13.18 Comparing cell data for tetrahedra and hexahedra. 144
13.19 The total number of molecules in the system versus time. 145
13.20 Equal potentials, $T = 2.5$, time = 10 146
13.21 Equal potentials, $T = 2.5$, time = 50 147
13.22 Equal potentials, $T = 2.5$, time = 90 148
13.23 Equal potentials, $T = 2.5$, time = 130 149
13.24 Equal potentials, $T = 2.5$, time = 170 150
13.25 Equal potentials, $T = 2.5$, time = 210 151
13.26 Different potentials, $T = 2.5$, time = 10 152
13.27 Different potentials, $T = 2.5$, time = 50 153

13.28 Different potentials, $T = 2.5$, time = 90 154
13.29 Different potentials, $T = 2.5$, time = 130 155
13.30 Different potentials, $T = 2.5$, time = 170 156
13.31 Different potentials, $T = 2.5$, time = 210 157
13.32 Equal potentials, $T = 1.0$, time = 10 158
13.33 Equal potentials, $T = 1.0$, time = 100 159
13.34 Equal potentials, $T = 1.0$, time = 200 160
13.35 Equal potentials, $T = 1.0$, time = 300 161
13.36 Equal potentials, $T = 1.0$, time = 400 162
13.37 Equal potentials, $T = 1.0$, time = 500 163
13.38 Different potentials, $T = 1.0$, time = 10 164
13.39 Different potentials, $T = 1.0$, time = 100 165
13.40 Different potentials, $T = 1.0$, time = 200 166
13.41 Different potentials, $T = 1.0$, time = 300 167
13.42 Different potentials, $T = 1.0$, time = 400 168
13.43 Different potentials, $T = 1.0$, time = 500 169
13.44 The distribution of molecules amongst 48 processors. 170

C.1 Finding velocities corresponding to a probability of $1/N$ in the Maxwellian velocity distribution. 202
C.2 Measured probabilities of maximum velocity in any timestep compared to prediction. 203
C.3 Variation of estimated neighbour list lifetime. 204

D.1 Key to colour coding of cells in example construction of referred cells. 208

E.1 A molecule referred across a boundary using a face centre/normal vector derived transformation. 229

F.1 Free path distribution measured by simulation. 233
F.2 Probability of finding a given number of other molecules simultaneously in a molecule's intermolecular potential. 234
F.3 Distribution of time spent in collision 235

List of Tables

1.1 Approximate sizes of substances relevant to biological applications 3

3.1 The merits of the identified modelling techniques judged against the criteria stated in section 3.1. 19

5.1 Reduced unit quantity expressions and reference values. 30

6.1 Data elements required by each zone in molConfigDict. 47
6.2 Pertinent molConfigDict entries for the three zone test case in figure 6.5. 53

11.1 Number of cells in test tetrahedral meshes generated by GAMBIT®. 110
11.2 The performance of molConfig for a cuboid system with varying aspect ratio. 112

12.1 Simulations comparing pr_04_1 ($r_{cut} = 2^{1/6}$) with gnemdFOAM. . 125
12.2 Simulations comparing pr_04_2 ($r_{cut} = 2.5$) with gnemdFOAM. . . 125

C.1 Experimental data and curve fitting parameters for $f_{v_N}(x)$. 202

Part I

Motivation and requirements

Chapter 1

Introduction

1.1 Research objectives

The ability to manipulate materials and create devices at the atomic level has been long anticipated [6], but has only recently started to become amenable to practical, widespread application. Amongst many of these rapidly emerging 'nanotechnologies,' fluid mechanics, for liquids in particular, plays an important but often poorly understood role. This lack of understanding is largely due to the difficulty of making direct experimental measurements at such tiny length scales. The aim of this research is to investigate, by computer simulation, the phenomena present in liquids at length and time scales relevant to nano-engineering systems in which liquids play a central role. This requires an understanding of how the behaviour of a liquid in a nanoscale geometry differs from that observed at the macroscale. The aim is to provide simulation and analysis methods to assist in the design of future devices that manipulate and exploit phenomena that are insignificant at the macroscopic scale.

The objectives of the work presented are to:

- establish the range of length and time scales, as well as physical phenomena (and their relative importance) relevant in the simulation and design of existing and future nano devices;
- devise and implement a simulation method for liquids suitable for the identified scales and phenomena, and capable of simulating realistic engineering geometries.

These are the enabling steps of a larger research project whose further objectives are to:

- compare simulations to published experimental data and the simulations of others, ensuring that the model converges to known results in simulations tending towards larger scales;
- carry out 'pathfinder' simulations of geometries suitable for future devices, with the intention of publication to allow experimental research groups to assess the accuracy of the predictions;
- provide a simulation code to academia and industry which is able to perform useful simulations to assist in understanding and designing nanoscale devices.

1.2 What is the 'nanoscale'

A defining characteristic of this research is that it is concerned with 'nanoscale' liquid flows. However, many of the application areas are speculative in nature, because their manufacturing and characterisation technologies are only just becoming feasible. Therefore the scale at which they will operate is undecided. Biological analysis is expected to be one of the primary uses of this research; therefore, to help establish a suitable range of sizes for simulation, table 1.1 gives the typical size of cells and molecules that may be manipulated.

Table 1.1: Approximate sizes of substances relevant to biological applications

Item	**Size**
Water molecule, OH distance	0.096nm
Amino acid	0.8nm
Cell wall pore	0.4-4nm
DNA: width of strand	2nm
DNA: length of helical turn	3.4nm
Quantum dots	2-9nm
Globular protein	4nm
Eucaryotic cell nucleus pore	10nm
Virus	20-100nm diameter
Bacteria	100-200nm to $\gg 1\mu$m
Red blood cell	6-8μm diameter

1.3 Nano-scale applications

The following describes some of the technologies, current and envisioned, where an enhanced understanding of the behaviour of small scale liquid flow would be beneficial.

- 'Lab-on-a-chip' devices can provide ubiquitous, rapid, cheap and disposable chemical and biological analysis and synthesis using minimal quantities of expensive or difficult to obtain samples or reagents. These devices can incorporate many individual component systems including pumps, reactors and separators for modification and manipulation of biological cells as well as mixers and transport channels where dispersion is to be minimised. These are currently microscale devices, but much of their behaviour depends on nanoscale processes, and the trend to further miniaturisation may make them nano devices in the future. The requirement for precision fluid manipulation at very small scales is fundamental to advances in this area of research; from [7]:

  > Central to the problem is the ability to move molecules over nanometre dimensions with high precision, selectivity, and temporal control; a capability which, when realised, will enable advances on both fundamental and technological fronts. Active control over transport of specific molecules at nanometre dimensions would comprise an enabling technology by permitting (a) new approaches to molecular separations that augment passive analyte-stationary phase interactions, (b) the study of reactions in which one or more reagents are available in extremely limited quantities, and (c) coupling of powerful methods of molecular identification, e.g. mass spectrometry, to spatially and temporally resolved molecular sampling methods; thereby permitting inherently small samples to be manipulated spatially.

  The ability to model fluid devices is a key feature in developments that will help to realise this capability. These devices often share common features:

  - transport and manipulation of complex suspended molecules with dimensions approaching those of the channel;
  - the actuation of flow and suspended particles is primarily electrokinetic: electroosmosis to create bulk motion of a fluid; electrophoresis

to move charged suspended particles; dielectrophoresis to move uncharged dielectric particles and electrowetting to move droplets across surfaces;

– multi-phase flows, either as immiscible droplets of liquid (for example, water droplets in oil) or liquid-gas systems.

- Designing and interpreting results from small scale instrumentation for measuring properties such as temperature, pressure, mass flow rate and species concentration. For example, nanoscale liquid chromatograph columns and detectors.

- Designing propulsion methods for 'free swimming' nanoscale devices and understanding the propulsion mechanisms of mobile micro-organisms.

- High throughput, highly selective filtering membranes, for large scale water purification, for example. Analysis of natural nanochannels (such as aquaporin proteins [8]) can provide inspiration for this.

- Electronics fabrication by photolithography takes place at the nanometre scale, involving the action of liquids on substrates. Understanding of this process is particularly relevant given the constant reduction in feature size (45nm gate length chips were released by Intel at the time of writing). MEMS and NEMS (Micro/Nano-Electro-Mechanical Systems) are also fabricated using similar technology. Nanoimprint lithography is an emerging fabrication technology where a droplet of liquid is placed on a substrate and flows into the nanoscale structure of a master template because of capillary forces. This allows "an entire wafer of circuits to be stamped out quickly and inexpensively" (from HP Labs[1]).

1.4 Physical phenomena

The main differences between macroscale fluid dynamics and that at the nanoscale are caused by physical effects that are absent or unimportant at larger scales. It is not possible to establish a hierarchy of importance of these effects because it would be application-specific and must be deduced by analysis and simulation in each case.

[1] http://www.hpl.hp.com/research/about/nanotechnology.html

1.4.1 Surface effects

Consider the surface area to volume ratio of a cube of side length, l, given by

$$\frac{6l^2}{l^3} \propto \frac{1}{l}.$$

As l decreases by many orders of magnitude to move from the macro to the nanoscale, the relative importance of the effects of surface interactions on the volume of fluid increase accordingly. The behaviour of the liquid becomes highly determined by surface effects and the details of fluid-solid intermolecular interactions. This gives rise to the dominance of surface tension, adsorption, molecular layering [9, 10], boundary slip and the effects of an electric double layer in a large portion of the volume of the system. These surface effects are inherently molecular in nature, specific to the substances involved and their state, are often too complex to encapsulate in simple phenomenological constants, and can rarely be predicted *a-priori*. This makes the behaviour of a fluid system dependant on multiple concurrent time and length scales. A simulation method is required to account for the small/short scale complexity, but still be amenable to providing insight over useful sizes and times.

1.4.2 Thermal fluctuations and diffusion

Thermal fluctuations are present in many properties of a small system, such as the local instantaneous temperature, density and pressure. They carry information about the state and thermodynamics of the fluid and drive diffusive processes, so they cannot be ignored. The time and length scales involved in nanoflows makes diffusion very important, particularly when trying to mix liquids together, because creating turbulent flow is normally not possible at small length scales. Small scale fluctuations also give rise to interesting, useful, but not fully understood processes such as thermodiffusion [11, 12]. At macroscopic length and time scales these fluctuations are not observed because they are fast and spatially small compared to the system, and are 'averaged out.'

1.4.3 Invalidation of continuum equations

It is possible to violate the continuum approximation of fluid mechanics or the normally assumed linear constitutive relationships. The former occurs often in

the vicinity of solid surfaces, where the molecular nature of the wall influences the structure of the fluid, causing it to form molecular layers. Where a channel is narrow enough that the whole cross section is affected by molecular layering, substantial departures from the predictions of continuum fluid dynamics occur [13, 14]. The latter can occur when a fluid is close to a solid surface or under high velocity, temperature or concentration gradients, leading to non-Newtonian, non-Fourierian or non-Fickian behaviour.

1.4.4 Electrokinetic effects

Any surface possessing a net electrical charge will give rise to an electrical double layer (EDL, also known as a Debye, or shielding layer) in an aqueous solution next to it [15, 16]. The surface charge attracts oppositely charged mobile electrolyte ions in solution, creating a concentration of charged ions near (and adsorbed onto) the surface. This region of liquid carrying a net charge can be acted upon by an imposed electric field, causing the charged liquid to move. This is known as electroosmotic flow (EOF).

The spatial extent of the EDL is dependent on the ion concentration and valency, pH and temperature of the solution, and is characterised by the Debye length, which is usually of the order of nanometres. This means that EDLs could cover entire flow regions of interest, especially for low electrolyte concentrations, rather than being confined close to channel walls, as in larger geometries.

Most applications for actuating small quantities of liquids will be solely electrokinetic (electroosmotic for an ionic liquid and electrophoretic for colloidal particles) and, as the system size reduces, this becomes increasingly true [7, 17–23]. From [18]:

> Electrokinetic 'pumping' is the leading technology for driving flows through microchannels, especially for channels where $h < O(10\mu m)$, because EOF can achieve much higher volumetric flow rates Q than pressure gradient-driven flow. In fully developed EOF, $Q \propto h$ for a given E [electric field strength]; in Poiseuille flow, $Q \propto h^3$ for a given pressure gradient. Electrokinetic pumping is also the leading technology for biochemical separations at the microscale, since the (nearly) uniform velocity profiles of EOF result in much less Taylor dispersion than Poiseuille flow. Taylor dispersion is a major limitation for compact biochemical separations using 'lab on a

chip' devices since efficient separation of large biomolecules requires high sample concentrations.

1.5 Thesis structure

The structure of parts I and II is the result of the following thought process:

- The objective is to study the fluid dynamics of nanoscale liquids by simulation.
  - What applications are there for this knowledge?
    - What literature is available to provide experimental results for comparison?
  - What physical effects are important at the nanoscale?
    - What simulation techniques are available to model these effects?
      - Hybridising molecular dynamics (MD) with continuum mechanics is the best option.
        - This requires an MD code that can operate in geometries defined by unstructured arbitrary polyhedral meshes with general, open boundaries, rather than the simple shapes with periodic boundaries offered by existing MD codes.
          - To achieve this the following parts of the MD code must be developed to operate in arbitrary geometries that have been spatially decomposed for parallel processing:
            - generation of initial configurations of molecules;
            - calculation of intermolecular forces;
            - tracking of the motion of molecules;
            - application of boundary conditions and driving forces;
            - measurement of spatially resolved properties.

Part III then describes the testing and performance of the MD code that has been developed.

Chapter 2

Experimental data literature

The papers discussed in this chapter are broadly categorised as being either concerned with practical nano-channels and nano-devices, or more fundamental studies of the details of how liquids behave near surfaces. The latter covers fluid slip, the surface force experiment, the presence of nanobubbles, and regions of reduced density near hydrophobic surfaces, topics which are largely interrelated. Only a representative sample of the literature is reviewed; papers offering compelling results or raising significant issues only are included.

Significant literature exists relating to flow in micro-channels. This is not covered here (see [24] for a review), except where either the behaviour is dependent explicitly on a nanoscale process, or the situation highlights a salient difference from what would be expected at the nanoscale.

There is a lack of high quality, repeatable experimental data to compare simulations to, so molecular dynamics (MD) simulations are also included because they can offer detailed insight into nanoscale phenomena. The MD simulations can be considered to be indirectly verified because the intermolecular potentials employed are either experimentally determined, or calculated from first principles using quantum mechanics.

2.1 Nano-channels and nano-devices

2.1.1 Nanotube and nanopipe flows

Simulations and experiments of flows in fullerene structured, single or multi-walled, crystalline nanotubes, have been reported for water and simple hydrocar-

bons [25–31]. In these studies, very fast filling dynamics for the fluid entering the tubes (fluid motion at speeds of hundreds of m/s over ps time scales) are reported, which do not agree with the macroscopic Washburn equation [32]. This is caused by virtually frictionless transport, due to the hydrophobic nature of the crystalline carbon. High speed fluid transport and the reactivity of carbon allows functionalisation of nanotube entrances and high throughput, selective and gated transport membranes to be created [33]. Other experiments directly observe the evaporation and condensation of water in nanotubes [34] and the effect of the presence of gas molecules in nanotubes has been simulated and qualitatively compared to experiments [35].

Nanopipes are much larger channels formed by chemical vapour deposition of material in pre-existing pores [25, 36]. These channels tend to have larger diameters than molecular nanotubes with walls comprising amorphous material rather than single crystals, and as such will have increased roughness. Flow experiments reported for these channels are in agreement with the Washburn equation.

2.1.2 Electrokinetic devices and flows

One of the main differences between nano and micro-channels is the size of the electrical double layer relative to the channel width. In nano-channels the double layers from opposite surfaces can overlap and result in a net charge of the liquid in the whole channel. This allows well controlled electrokinetic transport in these channels and the exclusion of one polarity of charged species [7, 17, 22, 23]

Experiments have been reported where the double layer structure has been controlled by surface chemistry and the application of external fields to create nanofluidic diodes [20] and transistors [19, 21].

2.2 Surface effects

The details of liquid-solid interfaces are very complex, depending on the species of each material and their thermodynamic state. Various experiments can be performed to attempt to characterise them [37], although the results of different techniques do not necessarily agree. The preparation and handling of samples as well as the measurement technique interact with the measurand.

The major area of investigation is the difference between hydrophobic and hydrophilic surfaces; these are important effects which are useful for controlling fluid behaviour, but their underlying mechanisms are not well understood. The situation is further complicated by water being the liquid of interest because it displays complex behaviour, usually attributed to subtle changes in local structure near a surface caused by its permanent dipole and hydrogen bond network.

2.2.1 Surface force experiment

The surface force apparatus [38] comprises two curved surfaces (made from molecularly smooth mica for example) which are piezoelectrically actuated to oscillate towards each other. The forces between them, due to interactions with a liquid or vapour, are measured. Films, coatings and treatments can be applied to the surfaces and the force data can be used to assess fluid-solid interactions. From this, the presence of fluid slip [39] (also see below) can be deduced and the behaviour of liquids, squeezed to be only several molecules thick, can be examined [40]. A good review of the apparatus, studies that have been conducted and the interpretation of results can be found in reference [41]. The resolution of typical instruments is approximately 0.1nm for the gap between the surfaces and 10^{-8}N for force on a surface.

2.2.2 Boundary slip

The no-slip velocity boundary condition used in macroscale fluid mechanics is known to be invalid in some rarefied gas flows [42] and the underlying mechanism is a long-studied problem [43]. Slip at liquid-solid interfaces is more complex and there is debate in the literature about its underlying mechanism; [44] presents a good review of the state of knowledge and range of studies.

Slip is often quantified using a slip length, δ, arising from Navier's hypothesis which states that the slip velocity, u_{slip}, is proportional to shear rate, $\dot{\gamma}$, i.e. $u_{slip} = \delta\dot{\gamma}$. A fixed value of slip length is considered to be an oversimplified representation with a limited range of applicability.

Hydrophilic surfaces do not generally display evidence of boundary slip so complex monolayers are typically used to make them hydrophobic. Published experiments are conducted with a range of fluids (water, dodecane, tetradecane and sucrose solutions) over different ranges of shear rate. Results split into studies

reporting long slip lengths, of the order of hundreds of nanometres to micrometres, or short slip lengths, of the order of tens of nanometres. In experiments reporting long slip lengths [45–51], the suggested mechanism is the presence of a layer of nano-bubbles forming a lubricating layer. It has been suggested [52] that long slip lengths are caused by the contamination of the surface with nanoscopic particles. Short slip lengths are reported in [52–54], where a region of depleted water density (a vapour layer) fits the observations.

Simulations of simple molecular dynamics systems with fluid impinging on molecularly smooth walls at very high rates of shear have been reported [10, 55] presenting short slip length results. The shear rates involved are much higher than would be seen in practical devices. Short slip length results have been obtained from low shear rate MD simulations using liquid decane [56] or simplified 10-atom chain molecules [57].

2.2.3 Nanobubbles

Several papers have been published describing the formation, morphology, stability and characteristics of 'nanobubbles': layers of vapour or gas that form on hydrophobic surfaces [58–61]. These are investigated using atomic force microscopy. Some authors are sceptical [62, 63] about whether these bubbles exist at all, or at least whether they are always present, or are nucleated by the measurement process. A recent study is careful to identify and exclude this possibility and show how the sample preparation technique can be controlled to produce or suppress bubble formation [61].

2.2.4 Density depletion

Neutron [62] or x-ray [63] reflectivity experiments have been performed to investigate the region of contact between water and a hydrophobic surface with high spatial resolution.

The experiments reported in [62], where deuterated water is in contact with a polished quartz surface which has a octadecyl-tricholorosilane (OTS) monolayer chemically attached[1], find that a 0.2-0.4nm wide region of depleted water density is found at the surface. In [63] the interface between de-ionised water and

[1] OTS and other silane monolayers are frequently used to produce hydrophobic surfaces; for example, see [52, 54, 61, 62, 64, 65].

"self-assembled methyl-terminated octadecyl chains of condensed octadecyltriethoxysiloxane (OTE)" deposited on a silicon oxide surface is examined. Similar results for the width and position of the depleted region to [62] are found.

Degassing the liquid samples, or saturating them with a gas other than air (argon and CO_2 for example [62]) has a significant impact on the results, as also seen in the nanobubble case [61].

Another experiment [64] used time-domain thermoreflectance to measure thermal conductivity at hydrophobic (created again using a self-assembled OTS monolayer) and hydrophilic interfaces, and found that only a very thin vapour layer (0.25nm) fits their data at the hydrophobic interface.

Molecular dynamics simulations of water in contact with a crystallised layer of long chain alkanes and alcohols (hydrophobic and hydrophilic respectively) are reported in [66]. No depleted density layer is observed at the hydrophilic surface, and a sub-nanometre depleted layer is observed at the hydrophobic surface. This paper demonstrates the ability to directly compare MD and x-ray experimental results. Molecular dynamics is also used to demonstrate some of the complexities of water in a high curvature hydrophobic 'pocket' [67]. The details of hydrogen bond rearrangement, drying, transient filling and emptying of the pocket are observed.

2.3 Summary

In nanoscale studies, significantly more consideration of the chemical composition of the fluid and device is required compared to the majority of fluid mechanics applications. Many authors stress the importance of controlled sample preparation and the characterisation of surfaces.

A wide variety of boundary conditions are possible because of different surface materials and roughnesses, the presence of nano-bubbles and slip, as well as the ability to form electric double layers. This can be regarded as an engineering opportunity for design flexibility, rather than properties that must be rigorously specified and simulated for a given material. To take advantage of this, nanofluidic devices can be designed by requiring a certain surface characteristic, with the intention of this demand being matched as closely as possible by fabrication techniques.

Chapter 3

Evaluation of simulation techniques

3.1 Specification and requirements

Four techniques have been identified as being potentially able to describe the flow of liquids at the nanometre scale: continuum CFD, molecular dynamics, the lattice Boltzmann equation, and dissipative particle dynamics. Modelling and simulation techniques will be evaluated against criteria that are divided into critical and desirable requirements.

Critical requirements. A modelling technique must:

1. have a computational cost that does not require excessive periods to simulate complex, realistic, 3D geometries over useful time scales, i.e. can provide solutions in a matter of hours or days on a small to medium size parallel computer;

2. be theoretically sound, i.e. derivable from basic physical principles, obeying the laws of conservation of mass, momentum, energy and the second law of thermodynamics[1];

[1]The second law of thermodynamics is only strictly applicable to systems of a minimum size over a minimum timescale. The equations of motion for molecules are time-reversible, but the production of entropy required to satisfy the second law means that, in general, the state of a whole system is not. This can be reconciled by treating the second law probabilistically, as formalised in the 'fluctuation theorem' [68–72].

3. not require the use of unknown or unknowable physical parameters, constants and properties to give accurate results;

4. have the ability to simulate all fluid mechanical phenomena, i.e. not be restricted to isothermal, adiabatic or isentropic processes;

Desirable requirements. It would be preferable for a modelling technique to:

1. be amenable to massive parallelisation to run on large supercomputers;

2. use existing, well established algorithms and software to facilitate rapid development;

3. be flexible enough to easily model complex phenomena and effects; for example multiphase and multispecies flows, suspended particles, chemical reactions or electrokinetic effects;

4. support multi-scaling and system size flexibility to allow a wide range of problem sizes and geometrical details to be concurrently analysed.

3.2 Continuum computational fluid dynamics

Continuum computational fluid dynamics (CFD), using the Navier-Stokes equations [73] and finite volume discretisation [74–76] is overwhelmingly the most widely applied fluid dynamics simulation technique in scientific and engineering applications. It is used to simulate suspended particles, combustion, chemically reacting systems, multiphase and electrokinetic flows, for example. The validity of the simulation depends on the validity of the governing equations used, and the accuracy depends on that of the boundary conditions and constitutive relations supplied (such as the relationship involving viscosity, thermal conductivity or heat capacity). As mentioned in section 1.4.3 and discussed further in the next chapter, the continuum approximation, therefore CFD's governing equations, can be invalid at very small scales.

In order to capture the effect of thermal fluctuations, the continuum equations may need to be stochastic differential equations [77–79], where the magnitude of the fluctuation is given by the fluctuation-dissipation theorem [80].

3.3 Molecular dynamics

Molecular dynamics simulates a fluid at a fundamental level by deterministically tracking the motion of every atom in a system, which move according to classical dynamics and interact with each other via specified intermolecular potentials. See section 5.1 for more details. The price paid for being able to offer a full microscopic description of a system is the high computational cost [81–83]. Systems of only a few tens of nanometres simulated for times of a few nanoseconds require hundreds or thousands of CPU hours.

3.4 Lattice Boltzmann

The Lattice Boltzmann (LB) method uses cellular automata techniques to simulate fluid mechanics, and is a "hyper-stylized version of the Boltzmann equation" [84]. It comprises a rigid lattice, over which computational particles move at one of a small set of discrete velocities and collide stochastically at lattice sites in order to reproduce hydrodynamic behaviour. Using the Lattice Boltzmann equation, the fluid density and velocity "can be shown to evolve according to the quasi-incompressible Navier-Stokes equations of fluid dynamics" [85] when the discrete velocities are chosen to conserve mass and momentum at each lattice step.

This method is particularly well suited to rapid computation (in 2 dimensions at least [84], p42) because it involves only a 'streaming' calculation to allow particles to move from one lattice position to another, and a 'collision' calculation at lattice vertices. This allows large systems with realistic geometries to be simulated in realistic time scales.

A recent paper [85] has compared LB and MD simulations for a "nontrivial nanoscopic fluid flow" — a 2D channel flow around an obstacle. While agreement with the MD simulation is demonstrated, it does not provide a rigourous demonstration of the claimed general applicability and valididity of LB. The MD simulation used to validate the LB simulation is relatively simplistic:

- the simulation is 2D: given that the authors are making quantitative computational speed comparisons between LB and MD, this is unrealistic;
- a purely repulsive Weeks-Chandler-Andersen (WCA) intermolecular potential is used, which is not a realistic model of a real fluid;

- the entire flow-field has been forced to be isothermal;
- the wall and obstacle atoms in the MD simulation are static, not undergoing thermal vibrations around lattice sites;
- there is no difference between wall-liquid and liquid-liquid intermolecular potentials.

The authors state that fluid momentum being transported by the flow velocity is a disadvantage:

> "The major advantages of LB over a purely hydrodynamic description are... highly irregular boundaries can be handled with ease because particles move along straight trajectories. This contrasts with the hydrodynamic representation, in which the fluid momentum is transported along complex space-time dependent trajectories defined by the flow velocity itself."

which seems to be a strange assertion. No mention of the energy equation or temperature profile for the LB simulation is made. The fluid viscosity and the velocity profile are both lattice spacing dependent. Comparison of the transient response of the LB and MD systems is not considered.

The theoretical basis of the Lattice Boltzmann technique being able to offer insight into microscopic processes in liquids (i.e. offer more insight than a Navier-Stokes solution) is questionable because of its derivation from the Boltzmann equation for dilute gases. The assumptions of molecular chaos and binary collisions only [86, 87] are essential to the Boltzmann equation and neither is appropriate for a liquid: molecular positions and velocities are highly correlated, and any molecule in a liquid will be in simultaneous 'contact' with approximately 50 others. This is acknowledged in [85]:

> "Although LB is a particle-based method, it cannot resolve the short ranged structural order of a dense liquid, since it describes a structureless lattice gas with no many-body interactions."

Furthermore, numerical artifacts and restrictions on the magnitude of spatial gradients, compressibility and heat transfer caused by the rigid lattice, and the strict timestep between collisions this imposes, are disadvantages [88].

3.5 Dissipative particle dynamics

Dissipative Particle Dynamics (DPD) (and the related Smoothed Particle Hydrodynamics, SPH) can be regarded as a method of coarse-graining molecular dynamics [89], where dissipative particles, representing a number of real molecules, move in an MD-like fashion, interacting with 'intermolecular' forces to reproduce hydrodynamic behaviour at length and time scales beyond the reach of molecular dynamics. It is possible to easily include coarse grained polymers in solvents to represent DNA, proteins or viruses for example.

There are two categories of simulation available which can be identified under the DPD (or SPH) description: 'soft' particles [90–92], similar to MD, or 'sharp' volume definitions which use a (dynamic) lattice structure such as a Voronoi tessellation. Voronoi tessellation is a method of discretising space around a set of points, such that a region is identified which is closer to the point it surrounds than any other. These points are regarded as dissipative particles, The construction of the tessellation itself forms part of the derivation of the dynamics of the system [93–98], where the fluxes across the mesh faces are calculated from some of the same information used to (dynamically) create the mesh. This technique may equivalently be seen as spatial discretisation of continuum equations, forming a Lagrangian, finite volume description. Implementations exist of Voronoi tessellation algorithms, *CGAL* [99] for example. Calculating the tessellation (which must be done at each timestep) is expensive in 3D, especially if solid walls are to be included [96], and parallelisation is difficult.

DPD or SPH are thermodynamically consistent, ensured by their adherence to the GENERIC framework [91, 93, 94] and they can easily incorporate thermal fluctuations [80, 91, 92, 95]. GENERIC (General Equation for the NonEquilibrium Reversible-Irreversible Coupling) is a framework for formulating time-evolution equations for non-equilibrium systems which ensures that any system with this structure is thermodynamically consistent, see [100–105].

3.6 Evaluation and comparison

Table 3.1 shows a (subjective) evaluation of the relative merits of the identified modelling techniques, judged against the criteria stated in section 3.1. The techniques have been ranked according to their overall suitability for nanofluidic

simulation.

Table 3.1: The merits of the identified modelling techniques judged against the criteria stated in section 3.1. Techniques are judged to be definitely suitable (✓), definitely unsuitable (×), or the answer is problem specific or unproven so far (?).

	CFD	MD	LB	DPD
Critical				
1	✓	×	✓	?
2	✓	✓	?	✓
3	×	✓	×	×
4	✓	✓	×	✓
Desirable				
1	✓	✓	✓	?
2	✓	✓	×	×
3	✓	✓	?	?
4	✓	×	×	✓
Rank	1	2	4	3

No technique fulfils all of the critical requirements. All techniques except MD fail critical requirement 3 because they need to be supplied with boundary conditions and constitutive models which, at the nanoscale, the behaviour of a system is very sensitive to. Providing that the intermolecular potentials are correct, MD avoids this problem because it simulates them directly. MD fails critical requirement 1 because it is orders-of-magnitude too computationally expensive to simulate realistically sized systems over timescales of the order of microseconds.

Continuum computational fluid dynamics and molecular dynamics can, between them, satisfy all of the simulation requirements, both critical and desirable. MD is able to accurately model the fluid properties, behaviour and boundary conditions, and CFD is able (given accurate boundary conditions and constitutive relations) to simulate large systems over useful timescales, except at the smallest of features, where MD is most applicable. Therefore, a simulation technique which hybridises continuum CFD and MD, each operating at their applicable scales, is the best option. This will be elaborated on in the next chapter.

Chapter 4

Hybrid molecular/continuum simulation

Successful fluid dynamics simulations using molecular dynamics have been reported [13, 14, 106–113], but MD is prohibitively computationally costly for simulations of systems beyond a few tens of nanometres in size, over timescales beyond a few tens of nanoseconds. Fortunately, the molecular detail of the full flow-field that MD simulations provide is often unnecessary; in liquids, beyond 5–10 molecular diameters ($\lesssim$ 3nm for water) from a solid surface the continuum-fluid approximation is valid and the Navier-Stokes equations with bulk fluid properties may be used [9, 40, 111]. Hybrid simulations have been proposed [83, 88, 114–116] to simultaneously take advantage of the accuracy and detail provided by MD in the regions that require it, and the computational speed of continuum mechanics in the regions where it is applicable. An example application of this technique is shown schematically in figure 4.1, where a complex molecule is electrokinetically transported into a nanochannel for separation and identification [23]. Only the complex molecule, its immediately surrounding solvent molecules, and selected near-wall regions require an MD treatment; the remainder of the fluid (comprising the vast majority of the volume) may be simulated by continuum mechanics. A hybrid simulation would allow the effect of different complex molecules, solvent electrolyte composition, channel geometry, surface coatings and electric field strengths to be analysed at a realistic computational cost.

A primary reason for choosing a hybrid approach is that less *a priori* knowledge is required of complex boundary and interface conditions and constitutive relations. It also provides the ability to unambiguously introduce complex phys-

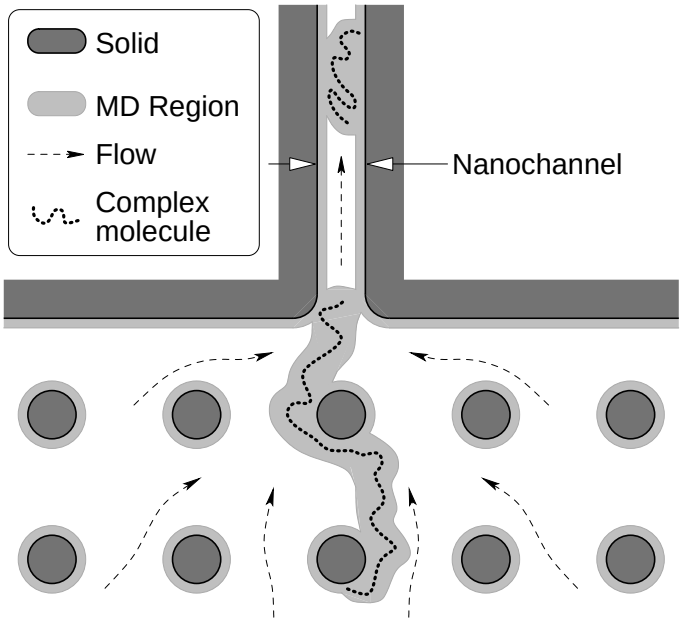


Figure 4.1: Schematic of an application of a hybrid MD/continuum simulation: complex molecules being transported into a nano channel. Only the complex molecules, regions near them and regions near solid surfaces need an MD treatment; the remaining volume can be simulated with continuum mechanics.

ical effects i.e. surfactants, proteins, reactions, dissolved gases, surface charges or electrokinetics. This contrasts with the lattice Boltzmann and dissipative particle dynamics methods which essentially provide alternative but equivalent means of solving the conventional continuum governing equations; they require user supplied boundary conditions and representative physical models which must be known *a-priori*. Hybrid simulations also allow direct integration of bulk hydrodynamic transport with more conventional MD applications, for example very high confinement or complex molecules [117] e.g. aquaporin proteins [118] or chemically gated nanotubes [33].

It should be noted that, for straightforward cases (simple fluids not under high confinement), a conventional Navier-Stokes description (including thermal fluctuations where necessary) with the no-slip velocity boundary condition (especially for hydrophilic surfaces) will be adequate; no MD simulation is required. The accuracy of such a simulation would depend on the accuracy of the transport properties used and their dependence on the state of the fluid. These can be determined in advance by MD [111].

The proposed hybrid technique uses the benefits of continuum simulation and MD in their respective domains of applicability and couples them together to

simulate across a wider range of length and time scales and physical effects than either is capable of individually. This technique is particularly appropriate for looking at nanoscale liquids, because this is where the continuum approximation breaks down and the molecular nature of the liquid starts to become significant. Two types of hybridisation can be employed:

1. coupled, domain decomposition hybridisation [83, 88, 114–116] where parts of the simulation domain are simulated by MD and other parts by CFD. The two components exchange information and both advance in time together;

2. performing a continuum simulation in the whole domain, but using MD at a set of sampling locations and times to derive accurate local boundary conditions and constitutive relations to guide and correct the continuum simulation. This is known as the heterogeneous multiscale method [119, 120]. This alleviates a major problem encountered in a fully coupled hybrid simulation: the disparity between simulation time scales. MD operates in the pico to nanosecond realm, and the continuum simulation may need to run for microseconds at least to simulate any real, useful systems. The results derived from the MD simulation can be stored and a library of results established, so that when a system encounters a particular flow condition and fluid state again, it could immediately reuse the result. This is similar to the In Situ Adaptive Tabulation (ISAT) [121] method used in chemical engineering control systems.

The two critical issues for a hybrid technique are:

- identifying breakdown of the continuum approximation or Navier-Stokes equations when flow conditions are sufficiently complex, non-continuum or non-equilibrium to require an atomistic treatment;

- coupling the MD and continuum domains together, ensuring both simulations are correct at the interface.

4.1 Breakdown of continuum equations

Breakdown of the Navier-Stokes (NS) equations can happen for two reasons:

- failure of the continuum approximation caused by the molecular nature of the fluid becoming important, this is seen near solid surfaces in particular;

- departure from hydrodynamic behaviour and the failure of linear constitutive relations (Newtonian viscosity, Fourierian heat conduction, Fickian diffusion) in bulk fluid, i.e. away from a wall, under high spatial gradients in velocity, temperature, density or concentration.

4.1.1 Continuum failure

The continuum approximation breaks down in liquids near solid surfaces when molecular effects are observed, such as discrete layering (oscillatory density profiles) [9, 10]. These effects are highly influenced by the details of the intermolecular forces and molecular structure of the surface in question [122]. It has been suggested by experiment [40, 122] and simulation [9, 10, 123] that this occurs within approximately five molecular diameters of a surface.

4.1.2 Deviation from hydrodynamics and linear response

In order to use the conventional continuum equations of fluid mechanics (the Navier-Stokes equations), the dynamics of the fluid must be in the hydrodynamic regime, this is where the variation of macroscopic variables occurs over length and time scales that are large compared to molecular scale events [124] (p219). In this case the assumption of local thermodynamic equilibrium is being implicitly made, where, although the macroscopic system is not in equilibrium (its properties vary in space and time) thermodynamic relationships hold in any small subdomain that is observed [124] (p220). In this case, linear-response theory [124] (p206) can be used to relate the flux of mass, momentum and energy to gradients in macroscopic variables in the liquid. In the case where spatial or temporal variations are too rapid, then the correlation between the behaviour of molecules (which decays away to zero in the hydrodynamic limit) means these relationships do not hold [124] (p241).

Non–Newtonian stress-strain behaviour can be expected when the strain rate, $\dot{\gamma} \gtrsim 2\tau^{-1}$, where τ is the characteristic molecular timescale of the liquid [55]. This corresponds to a velocity gradient of hundreds of ms^{-1}/nm for simple liquids, which is unlikely to be encountered in practical applications. It has been shown [125] that picosecond timescale events are necessary to violate the expected linear thermal behaviour. Simulations have shown that hydrodynamic continuum equations, when they have the correct transport coefficients, parameterised by the

local fluid state, are able to represent relatively large spatial gradients in temperature [111].

It is possible to encounter high concentration gradients, particularly where two fluids are mixing. In this case the diffusion behaviour may not be linear, or the diffusion coefficients might not be possible to know without molecular simulation.

4.2 Coupling MD and continuum regions

A theoretically sound and computationally efficient method is required to couple the continuum and atomistic regions of the simulation. There are two approaches to coupling [126]: the Schwarz method [83, 114], which couples the state of the continuum and atomistic regions, or coupling by passing fluxes of mass, momentum and energy across the interface between the regions [88, 115]. It is not clear whether either is universally more suitable.

Transferring information from the continuum to atomistic region is particularly challenging [114, 127–129] because a continuum flux of mass, momentum and energy must be discretised into an whole number molecules, which must have a realistic spatial distribution, and physically correct distribution of momentum and energy.

The transport properties used in the continuum domain must closely match those that arise from the intermolecular potential models used for the MD simulation [114, 127] in order to produce accurate results and stable coupling. Fortunately, deriving transport coefficients from MD is straightforward [111, 124, 130, 131]. An overlap region is usually required where both the continuum and atomistic simulation operate in order to ensure that both match at the interface.

4.3 Validation

Direct experimental validation of a nanoscale simulation is very difficult because experimental systems are complex, difficult to make accurate measurements of, and sensitive to small changes in setup. This does not diminish the value of the simulation however; as good a match as possible with experimental results should be obtained to establish confidence in the validity of the simulation. The simulation may then be used to probe the dependence of results on varying details of the

setup that cannot be well controlled or observed in the experiment. For example, in experiments where self-assembled monolayers are involved (see chapter 2) the coverage, quality and stability of the monolayers is critical, but difficult to control and characterise. Similarly, in reference [64] where heat conductance through a layered structure is involved, the sensitivity of the result to layer thicknesses or the quality of bonding between layers can be used to provide insight into the experimental results or control required in sample preparation.

Molecular dynamics simulations can be indirectly validated. If the intermolecular potentials used are accurate and valid for the state of the fluid and the MD code correctly calculates intermolecular forces and accurately integrates the equations of motion, then the simulation should not be substantially in error. An exception to this could occur when a system is close to an unstable state, i.e. a small perturbation or error could cause the possible solution trajectories to diverge. Also, when external fields or open boundary conditions are applied to the simulation, they may cause errors because the manner in which they act may be unphysical.

Part of a validation procedure will require that the MD component of the simulation reproduces results obtained from other MD codes. In addition, the hybridisation technique and the applicability of the continuum breakdown criteria may be probed by simulating a simple system (a channel flow for example) as a fully MD case, then introducing an increasing continuum region. If the hybridisation coupling technique is working correctly, the continuum CFD is valid and the MD region is large enough, then the results should stay the same.

4.4 Hybrid simulations in arbitrary geometries

Published studies [83, 88, 114–116] have demonstrated that hybrid simulations are viable, but these studies have dealt only with simple flows and domains. The geometries used have been typically simple cuboids with periodic boundaries. To be a useful, general engineering tool, a hybrid simulation must be able to simulate flows in complex 3D domains, where the geometry is derived from standard engineering CAD tools.

In order to achieve this, the MD component must be able to simulate flows in similar arbitrary geometries because, as can be seen in figure 4.1, the MD region of the domain will normally not be a simple shape. The remaining chapters describe

the creation of algorithms and a code to perform MD simulations in arbitrary, CAD derived geometries. This is an enabling step in creating a hybrid simulation tool, and also allows purely MD simulations in domains that were previously not possible.

Part II

Molecular dynamics in arbitrary geometries

Chapter 5

Molecular dynamics simulation overview and requirements

5.1 Molecular dynamics overview

Molecular dynamics simulates the deterministic, classical motion of a collection of molecules[1], which interact with each other according to an intermolecular potential, U. The equation of motion for a simple (rotationally symmetrical, no moment of inertia) molecule, i, is [130, 131]

$$m_i \mathbf{a}_i = \mathbf{f}_i, \tag{5.1}$$

where $\mathbf{a}_i$ is the acceleration vector of the molecule and $\mathbf{f}_i$ is the total force acting on it, given by

$$\mathbf{f}_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbf{f}_{ij}, \tag{5.2}$$

the sum of forces acting on molecule i caused by all other molecules j. If $U(r_{ij})$ is the potential energy between molecules i and j as a function of the intermolecular separation

$$r_{ij} = |\mathbf{r}_{ij}|, \tag{5.3}$$

[1] To preserve generality, 'molecule' will be used to mean 'atom', 'ion' or 'molecule', i.e. argon atoms, sodium ions, or water molecules would all be referred to as molecules. The physical context should indicate the meaning.

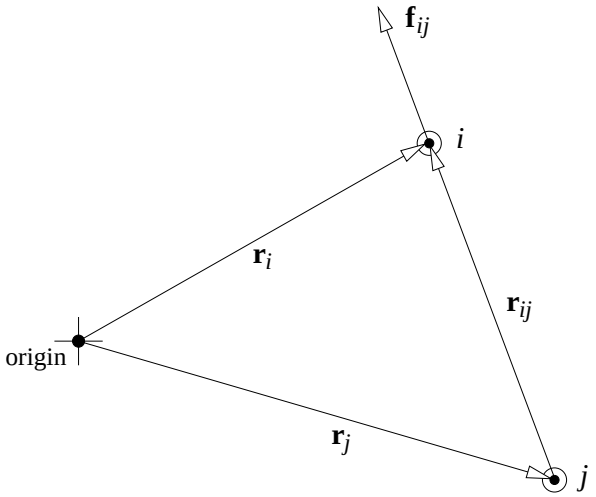


Figure 5.1: $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of molecules i and j, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $\mathbf{f}_{ij}$ is the force acting on i caused by j.

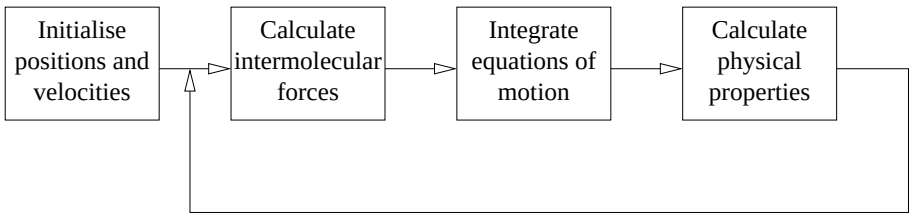


Figure 5.2: An overview of the molecular dynamics calculation process.

where

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j, \tag{5.4}$$

and $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of molecules i and j (see figure 5.1), then

$$\mathbf{f}_{ij} = -\nabla U(r_{ij}). \tag{5.5}$$

The key steps in a molecular dynamics simulation are shown in figure 5.2; each of these elements will be elaborated on in chapters 6 to 10. Calculating the intermolecular forces is overwhelmingly the most computationally time consuming step.

5.1.1 Reduced units

It is common in MD to perform numerical simulations using a set of scaled, or reduced units [130, 131]. Only three parameters are required to define the new units: length, energy and mass. Reduced units typically scale all quantities

Table 5.1: Reduced unit quantity expressions and the reference values used in this work. The reference length and energy are taken from the Lennard-Jones potential for argon (see section 5.1.3) and the reference mass is that of an argon atom. A reduced unit quantity can be converted to SI units by multiplying it by the value in the last column.

Defined quantities		
length	$l^* = l/\sigma_r$	$\sigma_r = 3.4 \times 10^{-10}\ m$
energy	$\epsilon^* = \epsilon/\epsilon_r$	$\epsilon_r = 1.65678 \times 10^{-21}\ J$
mass	$m^* = m/m_r$	$m_r = 6.6904 \times 10^{-26}\ kg$
Derived quantities		
force	$\mathbf{f}^* = \mathbf{f}\,(\sigma_r/\epsilon_r)$	$f_r = 4.87288 \times 10^{-12}\ N$
time	$t^* = t\sqrt{\epsilon_r/m_r\sigma_r^2}$	$t_r = 2.16059 \times 10^{-12}\ s$
position	$\mathbf{r}^* = \mathbf{r}/\sigma_r$	$r_r = 3.4 \times 10^{-10}\ m$
velocity	$\mathbf{v}^* = \mathbf{v}\sqrt{m_r/\epsilon_r}$	$v_r = 157.364\ ms^{-1}$
acceleration	$\mathbf{a}^* = \mathbf{a}\,(m_r\sigma_r/\epsilon_r)$	$a_r = 7.28340 \times 10^{-13}\ ms^{-2}$
temperature	$T^* = T\,(k_b/\epsilon_r)$	$T_r = 120\ K$
pressure	$P^* = P\left(\sigma_r^3/\epsilon_r\right)$	$P_r = 4.21530 \times 10^{7}\ Nm^{-2}$
mass density	$\rho_M^* = \rho_M\left(\sigma_r^3/m_r\right)$	$\rho_{Mr} = 1702.22\ kgm^{-3}$
number density	$\rho_N^* = \rho_N\sigma_r^3$	$\rho_{Nr} = 2.54427 \times 10^{28}\ m^{-3}$

towards unity. They may be used to perform a single simulation in a given reduced unit state, then obtain a range of values by scaling the simulation result.

Choosing the reference length and energy to be the same as those in the intermolecular potential being used, and the reference mass to be that of the molecules being simulated, simplifies and speeds up the calculation of intermolecular forces because those variables may be eliminated from the force equation. This work is intended to create a general framework for simulation with multicomponent fluids and mixed form potential functions, so the reduced units cannot be used to remove parameters from equations, merely to scale them. Defining a reference length, σ_r, energy, ϵ_r and mass m_r; reduced unit quantities have a superscript asterisk, i.e. l^*. The reduced unit of common quantities, as well as the reference value which a reduced unit should be multiplied by to convert it to SI, are shown in table 5.1 and are derived in appendix A.

5.1.2 Intermolecular potentials

Molecules interact with each other via a number of different physical mechanisms, although they are essentially all electromagnetic in nature [132, 133]. In this work only non-bonded interactions will be considered, which can be broadly categorised as short and long range forces:

Short range forces

When two molecules are sufficiently close, their electron clouds overlap, and, due to the Pauli exclusion principle, the electron density in the overlap region is reduced. This allows the positively charged nuclei, which are normally shielded by the electrons, to repel each other.

Long range forces

There are three principle mechanisms for long range forces.

- Electrostatic: when molecules have either a permanent charge or are neutral but have a permanent dipole (such as water) then the regions of the molecule with net charge experience Coulombic interactions. Electrostatic interactions are pairwise additive, i.e. the force between any pair of molecules is independent of the position or state of other neighbouring molecules.

- Induction: the electric field created by a charged or polar molecule distorts the electron charge distribution of neighbouring molecules, inducing a dipole in it, which then interacts attractively with the molecule that induced the distortion. Induction interactions are not pairwise additive because the distortion of the electron charge distribution of a particular molecule depends on the net electric field it experiences.

- Dispersion: The fluctuations of the electron distribution around molecules create instantaneous dipoles which interact to produce an attractive force between molecules that have no permanent charge or dipole. Dispersion interactions are not pairwise additive, but can be approximated as such.

In this work only short range repulsion and dispersion interactions (often referred to as 'Van der Waals' forces) between neutral, mono-atomic, spherically symmetric molecules have been considered, although the infrastructure has been created to allow the many more complex classes of interaction [130–133] that exist in the literature (for example reference [134] reviews the numerous potentials available for water and [135] offers a recent and promising [136] potential derived from first principles) to be easily added. It has been assumed that the forces between molecules are pairwise additive.

The equations used to describe short range potentials usually extend to infinite values of r_{ij}, however, it is necessary to impose a cut-off radius, r_{cut}, beyond which

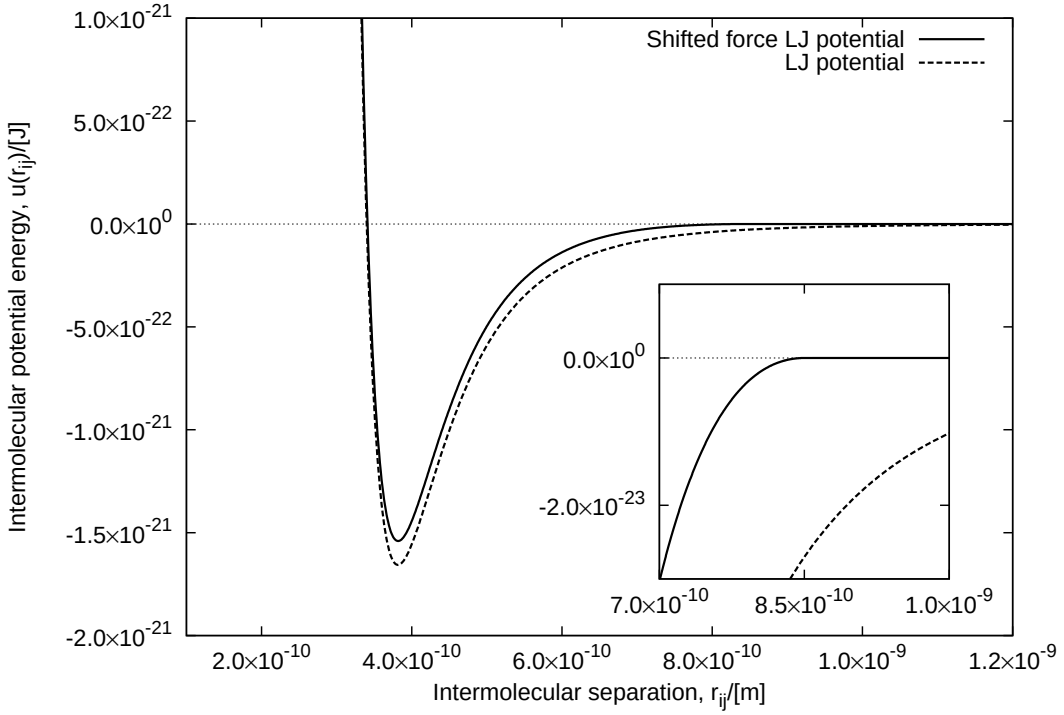


Figure 5.3: The Lennard-Jones (LJ) and shifted force LJ intermolecular potentials for argon. The inset figure shows that the shifted force version has a value and gradient of zero at 0.85nm, the cut-off radius, and is zero thereafter.

the potential energy is zero. A cut-off is required to reduce the computational cost of calculating the intermolecular forces, and the fact that potentials do not extend to infinity has a quantum mechanical basis [86, 137]. The cut-off radius is chosen where the value and gradient of the potential are both small.

5.1.3 The Lennard-Jones potential

Possibly the most famous potential used in molecular dynamics is the Lennard-Jones (LJ) 12-6 model:

$$U\left(r_{ij}\right) = \begin{cases} 4\epsilon\left(\left(\frac{r_{ij}}{\sigma}\right)^{-12} - \left(\frac{r_{ij}}{\sigma}\right)^{-6}\right), & r_{ij} \leq r_{cut} \\ 0, & r_{ij} > r_{cut}. \end{cases} \tag{5.6}$$

It was originally intended as a model for argon, but has become studied in its own right because of its simplicity (more accurate models for argon have existed for a long time [133]). Its equation of state has been thoroughly determined [138] and it can be used as a benchmark to check and compare simulation codes.

It is usually implemented as the shifted-force [131] Lennard-Jones potential:

$$U_{SF}(r_{ij}) = \begin{cases} 4\epsilon \left[\left(\left(\frac{r_{ij}}{\sigma}\right)^{-12} - \left(\frac{r_{ij}}{\sigma}\right)^{-6} \right) - \left(\left(\frac{r_{cut}}{\sigma}\right)^{-12} - \left(\frac{r_{cut}}{\sigma}\right)^{-6} \right) \right. \\ \quad \left. + \frac{12 r_{cut}\left(r_{ij} - r_{cut}\right)}{\sigma^2} \left(\left(\frac{r_{cut}}{\sigma}\right)^{-14} - \frac{1}{2}\left(\frac{r_{cut}}{\sigma}\right)^{-8} \right) \right], & r_{ij} \leq r_{cut} \\ 0, & r_{ij} > r_{cut}. \end{cases} \tag{5.7}$$

The shifted force potential ensures that the potential and its spatial gradient (hence the force) are zero at the cut-off radius, see figure 5.3. The shifted force LJ potential was used in all simulations in this work, with parameters which approximate argon unless otherwise stated: $\epsilon = 120 k_b$, where k_b is the Boltzmann constant, $\sigma = 0.34nm$ and $r_{cut} = 2.5\sigma_r \equiv 0.85nm$.

The equation for $\mathbf{f}_{ij}$ for the shifted force potential (derived in appendix B) in reduced units is

$$\mathbf{f}^*_{ij} = \frac{48\epsilon^*}{\sigma^{*2}} \left(\left(\frac{r^*_{ij}}{\sigma^*}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{ij}}{\sigma^*}\right)^{-8} - \frac{r^*_{cut}}{r^*_{ij}} \left(\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-8} \right) \right) \mathbf{r}^*_{ij}. \tag{5.8}$$

5.1.4 Periodic boundaries

Molecular dynamics simulations are typically performed with periodic boundary conditions [124, 130, 131] where the system is simulated as if it is surrounded by an infinite number of copies of itself. In practical terms, this means that when a molecule leaves one side of the domain, it is 'wrapped-round' and reintroduced on the other side. Molecules calculate forces with periodic images of the system, which are actually molecules residing across periodic boundaries on the other side of the system, see figure 5.4.

5.2 Limitations of conventional molecular dynamics

5.2.1 System geometry

Existing, widely used MD codes (such as LAMMPS, NAMD, AMBER, GROMACS or DL_POLY) can only simulate systems represented by simple domains:

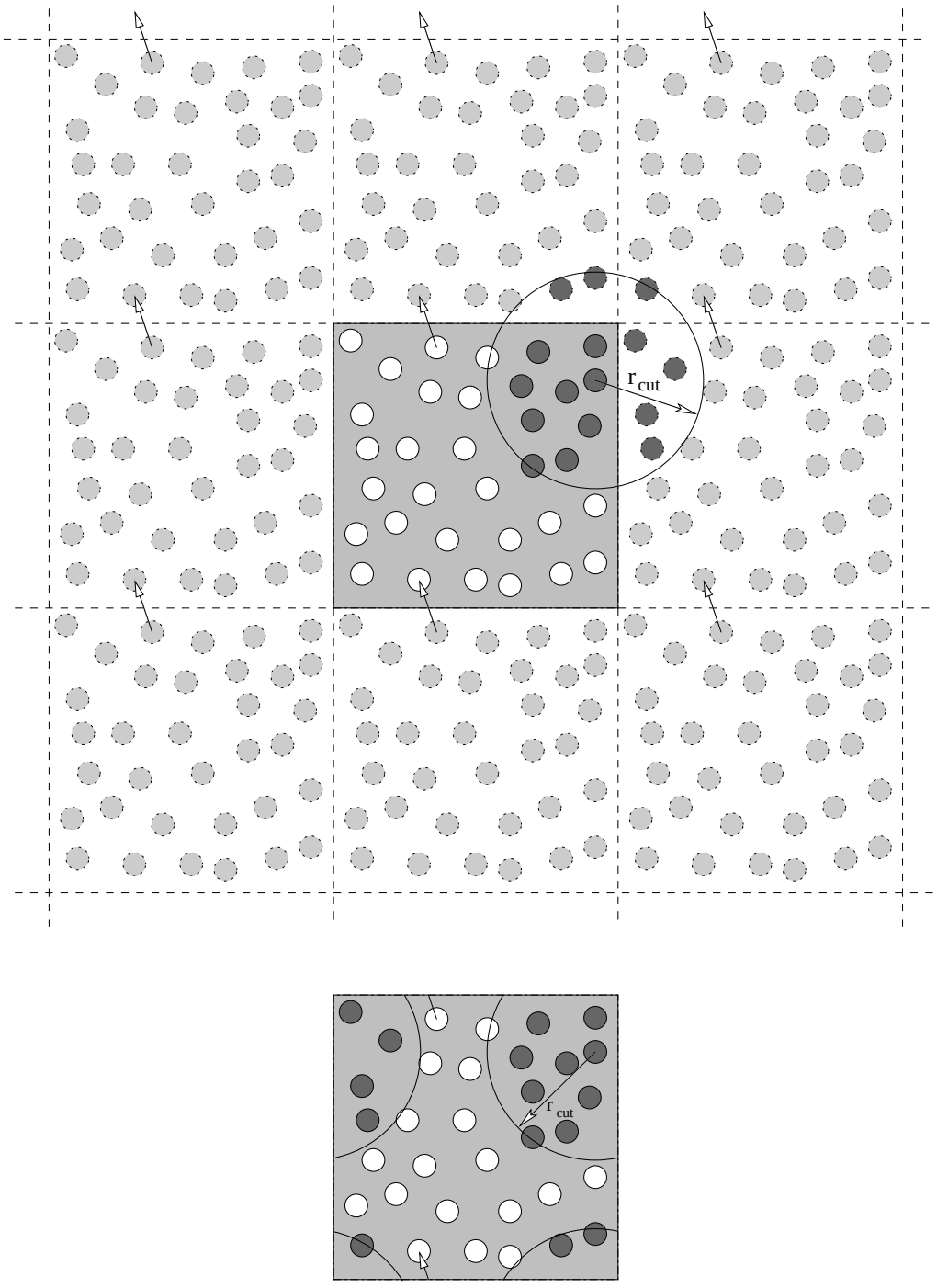


Figure 5.4: Periodic boundaries. **Above:** the notional arrangement. The periodic system of molecules (centre) is surrounded by images of itself. When a molecule leaves the system it enters one of the images, and the image on the opposite side of the system moves a corresponding molecule in. A molecule calculates intermolecular forces between all molecules within r_{cut} residing in the system and neighbouring images. **Below:** the computational implementation. Molecules leaving the system are 'wrapped-round' and re-enter from the opposite side. Intermolecular forces are calculated with molecules positioned on the opposite side of the domain.

volumes that are space filling when periodic boundaries are applied, normally cubes, cuboids, or parallelepipeds. This is because most MD simulations are intended to examine a system in an infinite, unbounded medium, without the influence of solid surfaces.

5.2.2 Neighbour lists

The conventional method of MD force evaluation in distributed parallel computation is to use the cells algorithm to build neighbour lists for interacting pairs [130, 131]. The *replicated molecule* method provides interactions across periodic boundaries and interprocessor boundaries, where the system has been spatially partitioned [139, 140].

The spatial location of molecules in MD is dynamic, and hence not deducible from the data structure that contains them. A neighbour list defines which pairs of molecules are within a certain distance of each other and so need to interact via intermolecular forces. When considering systems where the geometry is defined by a mesh of unstructured, arbitrary polyhedral cells, that may have been divided into irregular and complex mesh portions for parallel processing, neighbour lists have two limitations:

1. **Interprocessor molecule transfers:** A molecule may cross an interprocessor boundary at any point in time, even part of the way through a timestep. At this point it should be deleted from the processor it was on and an equivalent molecule created on the processor on the other side of the boundary. Given that neighbour lists are constructed as lists of array indices, references or pointers to the molecule's location in a data structure, deleting a molecule would invalidate this location and require searching to remove all mentions of it. Likewise, creating a molecule would require the appropriate new pair interactions to be identified. Neither is practical due to the computational cost involved. It is conventional to allow molecules to stray outside of the domain controlled by a processor and carry out interprocessor transfers (deletions and creations) during the next neighbour list rebuild. This is straightforward when the spatial region associated with a processor can be simply defined by a function relating a position in space to a particular cell on a particular processor (i.e. a uniform, structured mesh, representing a simple domain). In a geometry where the space in question

is defined by a collection of individual cells of arbitrary shape, this is more difficult. For example, the location the molecule has strayed to may be on the other side of a solid wall on the neighbour processor, or across another interprocessor boundary.

2. **Spatially resolved flow properties:** MD simulations used for flow studies must be able to spatially resolve fluid mechanical and thermodynamical fields. This is achieved by accumulating and averaging measurements of the properties of molecules in individual cells of the same mesh that defines the geometry. If a molecule is allowed to stray outside of the domain controlled by a processor, as above, then it would not be unambiguous and automatic which cell's measurement the molecule should contribute to.

Both of these problems could be mitigated by communicating with the neighbouring processor to determine and communicate which cell a molecule outside the domain should be in, or alternatively maintaining a local copy of the geometry of the neighbouring processors to a certain depth. In this work, however, either of these options would result in an inflexible arrangement, with each additional simulation feature requiring special treatment; neighbour lists have therefore not been used.

Neighbour lists have some unfavourable features that limit their computational speed in large and non-equilibrium simulations. If one region of the fluid has a high temperature or high bulk velocity, then the high molecular velocities will cause the neighbour list to be invalidated and rebuilt often, limiting the calculation speed in the whole domain. Increasing the size and temperature of a system also inherently reduces the number of timesteps that a neighbour list is valid for, thereby increasing the computational cost. This relationship can be predicted, see appendix C.

A new algorithm has been designed either to replace the neighbour list technique outright, or to construct the neighbour list for simulations operating on meshes of unstructured arbitrary polyhedral cells.

5.3 Requirements for arbitrary geometry simulation

In order to produce a useful, general simulation tool for hybrid simulations, the MD component must be able to model complex geometrical domains represented by unstructured mesh geometries of arbitrary polyhedra (such as those generated by large scale, automatic meshing of geometries created by engineering CAD tools) that have been parallelised for distributed computing by spatial decomposition. This is also useful functionality in its own right; MD simulations of complex nano-devices derived from CAD models can be made directly by performing 'CFD with molecules'. To achieve this:

- initial configurations of molecules corresponding to volumes defined by the mesh must be generated. The algorithms underpinning a preprocessing tool able to create such configurations are described in chapter 6. This is able to fill volumes defined by a zone of the mesh (a set of cells) with a single species crystal lattice of molecules. The user may specify the lattice structure, orientation, density, temperature and average velocity;

- intermolecular forces must be calculated, taking account of periodic and interprocessor boundaries. This is the most important and computationally demanding aspect of any MD simulation and methods are detailed in chapter 7 and appendices D and E;

- molecules must be tracked as they move through the mesh from cell to cell (chapter 8);

- boundary conditions and constraints must be imposed to control the state and define the dynamics of the system. Chapter 9 discusses some simple methods that have been implemented and outlines proposals for more powerful techniques;

- spatially resolved measurements of the properties of the fluid and the flow must be made. Methods for measuring and temporally averaging properties in individual cells are given in chapter 10 for a heterogeneous fluid containing an arbitrary number of species.

5.3.1 Parallelisation

The simulation is to be parallelised by domain decomposition [139, 140], where the simulation volume is divided into smaller portions and each processor is given responsibility for simulating the molecules residing in its portion. Molecules move between processors when they enter and leave a portion. Where the simulation domain is simple (a cube or a cuboid for example) then the decomposition is straightforward (divide into smaller cubes or cuboids), which is what is done by existing MD codes. Where the domain is a complex shape constructed using unstructured arbitrary polyhedra, the volume must be decomposed into irregular portions. There are libraries available for partitioning meshes (METIS [141] for example) which minimise interprocessor connections, but can produce unintuitively shaped portions over which the user does not have much direct control.

Intermolecular forces must be passed across processor boundaries, and where the decomposition into portions can be entirely arbitrary, processors that do not share a boundary still may need to communicate.

5.4 Implementation in OpenFOAM

All algorithms described have been implemented in OpenFOAM [5], which is an open source C++ library intended for continuum mechanics simulation of user-defined physics (primarily used for CFD) in arbitrary, unstructured geometries. See appendix G for an overview. The MD simulation code (see chapters 7 to 10) has been built using OpenFOAM's lagrangian particle tracking library (see section 8.3) and is called gnemdFOAM. All features of OpenFOAM have a common infrastructure for distributed memory parallel processing using MPI for communications.

The initial molecule configuration generation tool for gnemdFOAM is also written using OpenFOAM and is called molConfig, see chapter 6. molConfig operates independently on individual portions of a mesh that have been spatially decomposed to run in parallel, allowing systems comprising very large numbers of molecules to be created because they never need to all be contained in the memory of a single computer. The molecular configurations are the same whether generated in parallel or in serial; crystal lattices generated in parallel are continuous and defectless across interprocessor boundaries. All parallel decomposition and reconstruction is dealt with by OpenFOAM using existing functionality.

5.4.1 Mesh description

The mesh in OpenFOAM is flexible and powerful: it is unstructured and built from arbitrary polyhedra. From the OpenFOAM user guide [5]:

> "By default OpenFOAM defines a mesh of arbitrary polyhedral cells in 3D, bounded by arbitrary polygonal faces, i.e. the cells can have an unlimited number of faces where, for each face, there is no limit on the number of edges nor any restriction on its alignment."

A list of mesh vertex positions is stored and a list of mesh faces is constructed; each face is an ordered list of vertex numbers. Cells are constructed as a list of face numbers. Vertices may be shared by several faces and cells. Cells can be grouped together into zones, each zone representing a region of the domain with common characteristics. Zones are used in molConfig to define regions to be filled with different crystals.

Patches

Patches are a set of OpenFOAM classes. Each patch comprises a collection of cell faces representing a mesh boundary of some form — they may provide solid surfaces, inlets, outlets, symmetry planes, periodic planes, or interprocessor connections.

5.4.2 Continuum solver integration

One of the main advantages of using OpenFOAM is that it offers tight integration with highly capable continuum mechanics solvers using the same geometry and underlying numerics as the molecular component. This facilitates hybridisation greatly. It also allows the extension of physical phenomena which may be incorporated: for example Maxwell's equations can be solved on the same mesh as the molecules occupy to apply an external electromagnetic field, or to apply the mean-field value of long range electrostatic interactions.

5.4.3 Reduced unit implementation

Currently gnemdFOAM and molConfig are implemented in reduced units, as are all quantities that the user must specify. In future versions all user interface

quantities will be in SI units and a set of reduced units will be used for internal calculation only.

Chapter 6

Initial molecular configuration generation

In order to initialise an MD simulation, molecules must be placed in the zones of a mesh. In conventional MD simulations this is typically done by filling the system with a homogeneous lattice, with any additional features, such as complex molecules or solid boundaries, being placed by hand. The tool molConfig has been created to perform 'flood fills' of zones of cells in the mesh with lattices of molecules. The zones may have arbitrary shape and be distributed across any number of processors. Lattices of molecules may be used to fill zones containing solids, liquids or gases: the intermolecular potential, temperature and density of the molecules will determine if the ordered lattice state is preserved (for a solid) or if it quickly collapses or melts into a random, fluid-like spatial distribution.

The algorithm used by molConfig fills any volume defined by a zone of cells with a single lattice of molecules of specified orientation and alignment. It generates an expanding cube of lattice unit cells, starting from a user-specified position (known as an 'anchor' for the lattice), and angled according to a user-specified orientation (see figure 6.1). Each of these unit cells is populated with molecules corresponding to a user-specified type of lattice for the zone. Each proposed molecule position is tested to see which cell it occupies[1]. If this cell comprises part of the zone currently being filled, the molecule is accepted; if not, the molecule is rejected. Using lattice unit cells makes the generation of the expanding cube independent of the specific crystal structure. Additional layers of unit cells are

[1]Finding which cell a position corresponds to is an existing function in the mesh description in OpenFOAM.

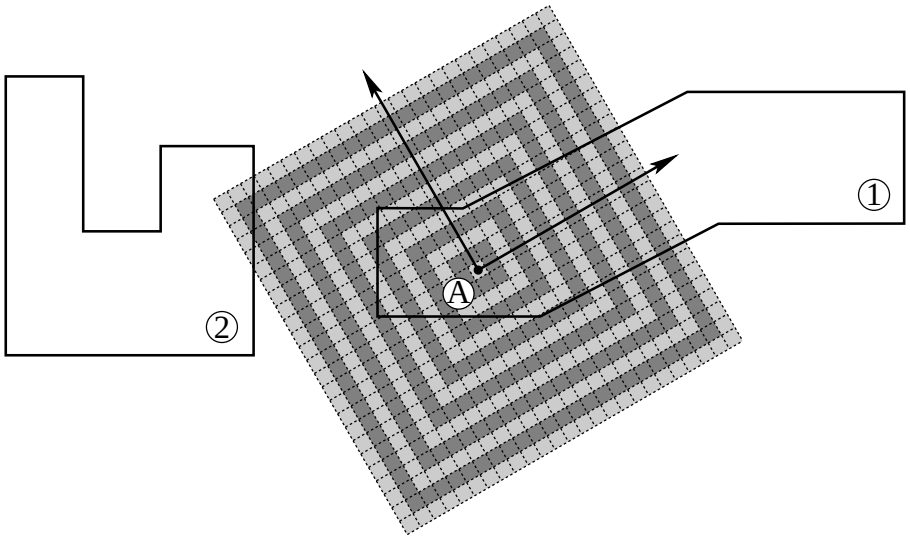


Figure 6.1: A single mesh zone comprising two disconnected volumes. These volumes are filled by creating an expanding cube of lattice unit cells from the anchor point, **A**. Volume 2 will be filled as long as volume 1 is not fully filled before the expanding cube reaches the edge of volume 2.

added to the cube until no molecules are accepted for placement in a complete layer, meaning that the volume has been completely filled. An exception is made if no molecules have been placed in the zone so far, enabling the lattice anchor to be placed outside the volume of the zone.

Concave and disconnected volumes of cells belonging to the same zone must be able to be filled (see figure 6.1) which makes more sophisticated algorithms (for example projecting rays to find the extents of the domain) difficult to implement reliably. The ability to fill disconnected volumes allows, for example, wall regions with gaps for fluid inlets to be filled with a single lattice. It is also a requirement for parallel processing because mesh decomposition will often produce disconnected volumes for a zone that is fully-connected in the undecomposed mesh.

6.1 Creating an expanding cube of unit cells

Consider a solid cube of unit cells comprising n layers of concentric hollow cubes of one unit cell thickness, starting with $n = 0$ as a single unit cell at the centre. A layer of this expanding cube is constructed as a combination of a top and bottom 'cap' plus 'rings' of cells, see figure 6.2. A Cartesian lattice coordinate system

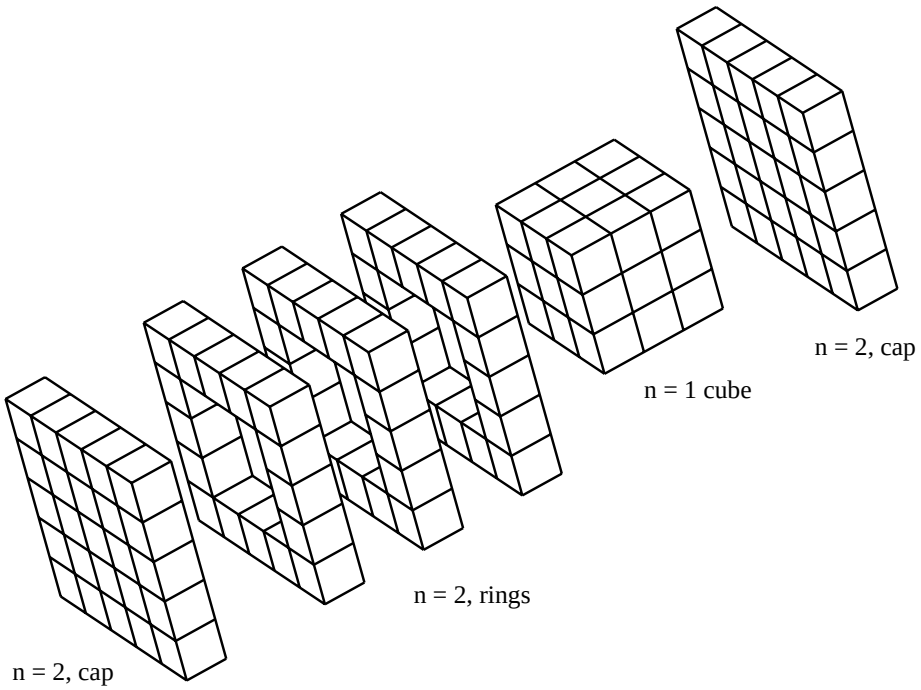


Figure 6.2: Exploded view of the unit cells in layer $n = 2$ added to the cube of cells containing layers $n = 0$ and 1. A $(2n + 1) \times (2n + 1)$ square of unit cells — a 'cap' — is placed on each end of the cells from the previous layer and $(2n - 1)$ rings are added between the caps to establish the next full layer of the cube.

local to the expanding cube is defined to specify the position of a unit cell,

$$\mathbf{\Lambda} = \Lambda_x \mathbf{x}_\lambda + \Lambda_y \mathbf{y}_\lambda + \Lambda_z \mathbf{z}_\lambda \equiv (\Lambda_x, \Lambda_y, \Lambda_z)\,,$$

where Λ_x, Λ_y and Λ_z are integers for unit cell centres and $\mathbf{x}_\lambda$, $\mathbf{y}_\lambda$ and $\mathbf{z}_\lambda$ are the lattice coordinate system unit vectors.

Caps Caps are placed in the xy lattice coordinate plane. The top cap has $\Lambda_z = n$ and the bottom cap $\Lambda_z = -n$. For each cap, a complete square is generated by setting

$$\Lambda_y = \{-n, -n+1, \ldots, n\}\,,$$

and for each value of Λ_y generate a line of cubes by setting

$$\Lambda_x = \{-n, -n+1, \ldots, n\}\,.$$

The centre unit cell ($n = 0$) is a special case and is placed at the lattice coordinate origin.

Rings There are $2n - 1$ rings, with

$$\Lambda_z = \{-n + 1, -n + 2, \ldots, n - 1\} .$$

The rings are created by a 2D expanding layer of squares algorithm. The (Λ_x, Λ_y) coordinates of the unit cells to be added to the layer of the ring are generated by adding a layer to a square of unit cells. Unit cells are generated along one side of the square and replicated around the other three sides to create the whole layer. Note that this replication cannot be performed by simply rotating or reflecting the positions of molecules on one side to create the other three, because the unit cells are generally not rotationally symmetric.

The generation of an expanding square of unit cells is shown in figure 6.3. The first cell ($n = 0$) is a special case and is placed at the origin of the lattice coordinate system. The coordinates of a unit cell will be given by n, the layer it is in, and r, the repetition counter: $r = \{0, 1, 2, \ldots, 2n - 1\}$, which generates the correct number of unit cells along each side. $\mathbf{\Lambda^k}$ represents the coordinates of the unit cells on the k^{th} side of the layer, $k = \{1, 2, 3, 4\}$:

$$\mathbf{\Lambda^1} = (n, -n + (r + 1)) , \tag{6.1}$$

$$\mathbf{\Lambda^2} = (n - (r + 1) , n) , \tag{6.2}$$

$$\mathbf{\Lambda^3} = (-n, n - (r + 1)) , \tag{6.3}$$

$$\mathbf{\Lambda^4} = (-n + (r + 1) , -n) . \tag{6.4}$$

Note that the coordinate of each successive side is generated by swapping the x and y coordinates, negating the $y \rightarrow x$ transfer, i.e.

$$\mathbf{\Lambda}^{k+1} = \left(-\Lambda_y^k, \Lambda_x^k\right) . \tag{6.5}$$

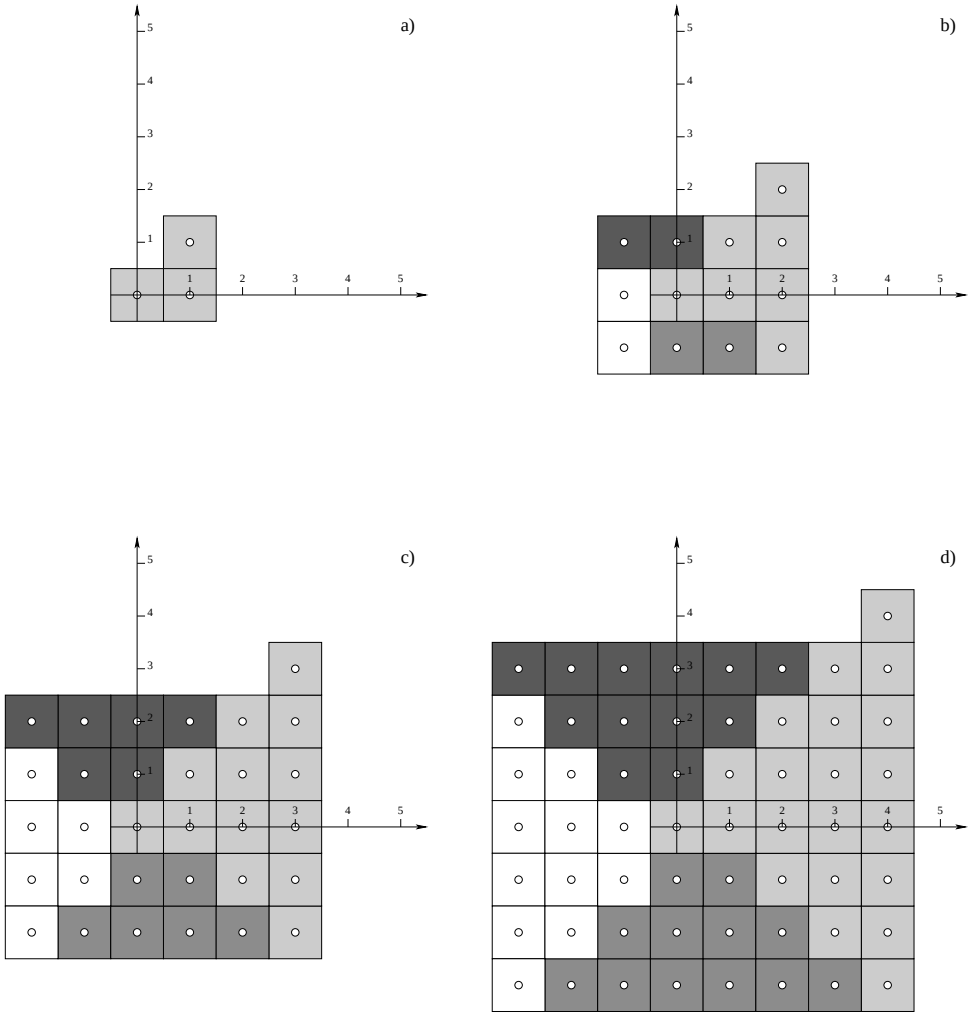


Figure 6.3: The construction of the sides of the square for layers $n = \{1, 2, 3, 4\}$. $\mathbf{\Lambda^1}$: vertical, positive x, light-grey. $\mathbf{\Lambda^2}$: horizontal, positive y, dark-grey. $\mathbf{\Lambda^3}$: vertical, negative x, white. $\mathbf{\Lambda^4}$: horizontal, negative y, mid-grey. a) $\mathbf{\Lambda^1}$ for $n = 1$; b) all sides for $n = 1$ and $\mathbf{\Lambda^1}$ for $n = 2$; c) all sides for $n = 2$ and $\mathbf{\Lambda^1}$ for $n = 3$; d) all sides for $n = 3$ and $\mathbf{\Lambda^1}$ for $n = 4$.

6.2 Zone specification: molConfigDict

For each zone of cells in the mesh, the user must specify the details of the lattice to be created. This data is supplied through the molecule configuration dictionary[2] file: molConfigDict. The entry for a zone contains the items shown in table 6.1.

The overall density of molecules placed in the zone (the total number of molecules divided by the total zone volume) may only be approximately correct. It will deviate slightly from the bulk value specified unless the region dimensions correspond to an integer number of unit cells and the anchor and orientation angles place and align the lattice so that the domain is filled accordingly.

It is possible to generate molecular data from an underlying, spatially-varying data field. For example, a temperature and velocity field from an initial continuum fluid mechanics simulation of the geometry using OpenFOAM may be available, and the molecular velocities can be generated using the local values of temperature and bulk velocity.

To create a solid wall in a simulation it is often useful to tether molecules into a lattice, for example using a spring potential [110, 142], so that they retain their structure. If tethered is *yes*, then the initial locations of molecules in this zone will be their tether positions.

In future developments of the utility, realistic crystals will be available in a library. The id, mass and density entries will not be necessary, and additional information may be required. For example, for latticeStructure = *NaCl*, molecules would be placed with the correct structure and spacing for the specified temperature (and possibly pressure), and given ids of *Na* and *Cl*, with the appropriate mass and charge assigned to each.

6.3 Generating molecule positions from a unit cell

A rank two rotation tensor, $\mathbf{R}$, is created using the ϕ, θ, ψ convention Euler angles [143] to specify the orientation of the lattice relative to the global Cartesian

[2] All user specified data is supplied to OpenFOAM via dictionary files.

Table 6.1: Data elements required by each zone in molConfigDict.

Entry	Description	Data Required
density	bulk number density	scalar
temperature	initial temperature	scalar
velocityDistribution	random number distribution (at the specified temperature) to choose initial velocities of molecules from, see section 6.6	*uniform* or *maxwellian*
bulkVelocity	average velocity to add to random velocity component	vector
id	identification of type of molecule to be added — used to determine which intermolecular potential to use	string, e.g. *LJ*
mass	mass of each molecule to be placed	scalar
latticeStructure	what structure of lattice to create	string, e.g. *SC*, *BCC* or *FCC*
anchor	the position specifying the starting point of the lattice	vector
anchorSpecifies	specifying whether the anchor is the position of a molecule or the corner of a unit cell	*molecule* or *corner*
tethered	are the molecules to be tethered to their initial positions?	*yes* or *no*
orientationAngles	the orientation of the lattice, see section 6.3	triplet of angles

coordinate system:

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \tag{6.6}$$

where,

$$\begin{aligned} r_{11} &= cos(\psi)cos(\phi) - cos(\theta)sin(\phi)sin(\psi), \\ r_{12} &= cos(\psi)sin(\phi) + cos(\theta)cos(\phi)sin(\psi), \\ r_{13} &= sin(\psi)sin(\theta), \\ r_{21} &= -sin(\psi)cos(\phi) - cos(\theta)sin(\phi)cos(\psi), \\ r_{22} &= -sin(\psi)sin(\phi) + cos(\theta)cos(\phi)cos(\psi), \\ r_{23} &= cos(\psi)sin(\theta), \\ r_{31} &= sin(\theta)sin(\phi), \\ r_{32} &= -sin(\theta)cos(\phi), \\ r_{33} &= cos(\theta). \end{aligned}$$

Another rank two tensor, $\mathbf{G}$, is required to specify the scaling of the cubic unit cells to the shape of the lattice,

$$\mathbf{G} = \begin{pmatrix} g_{11} & 0 & 0 \\ 0 & g_{22} & 0 \\ 0 & 0 & g_{33} \end{pmatrix}. \tag{6.7}$$

This is specific to the lattice structure used and the density of the crystal. $\mathbf{R}$ and $\mathbf{G}$ are used in conjunction with the lattice anchor, $\mathbf{A}$, to transform a position in lattice coordinates, $\mathbf{\Lambda}$, to a position, $\mathbf{P}$, in the global coordinate system

$$\mathbf{P} = \mathbf{A} + \mathbf{R} \cdot (\mathbf{G} \cdot \mathbf{\Lambda}). \tag{6.8}$$

Each lattice type will place its own number of molecules in the appropriate positions around the unit cell centre. Three examples given below are the simple, body-centred and face-centred cubic lattices where

$$g_{11} = g_{22} = g_{33} = g,$$

because the lattices are cubic. These examples are based on lattice generation steps in [130].

Simple cubic (SC)

For ρ, the user specified bulk number density,

$$g = \rho^{-1/3}.$$

and if anchorSpecifies = *molecule* then a single molecule is placed at the unit cell centre, $\mathbf{\Lambda}$. If anchorSpecifies = *corner* the molecule is shifted in the lattice coordinates to $\mathbf{\Lambda} - (0.5, 0.5, 0.5)$ prior to transformation by equation (6.8).

Body centred cubic (BCC)

For a unit cell centre $\mathbf{\Lambda}$, if anchorSpecifies = *molecule*, molecules are placed at

$$(\Lambda_x, \Lambda_y, \Lambda_z)\,,$$
$$(\Lambda_x + 0.5, \Lambda_y + 0.5, \Lambda_z + 0.5)\,,$$

in lattice coordinates, then transformed to the global coordinate system using equation (6.8), where

$$g = \left(\frac{\rho}{2}\right)^{-1/3}.$$

If anchorSpecifies = *corner* then the two positions above are shifted in the lattice coordinates by $(-0.25, -0.25, -0.25)$ prior to transformation.

Face centred cubic (FCC)

For a unit cell centre $\mathbf{\Lambda}$, if anchorSpecifies = *molecule*, molecules are placed at

$$(\Lambda_x, \Lambda_y, \Lambda_z)\,,$$
$$(\Lambda_x, \Lambda_y + 0.5, \Lambda_z + 0.5)\,,$$
$$(\Lambda_x + 0.5, \Lambda_y, \Lambda_z + 0.5)\,,$$
$$(\Lambda_x + 0.5, \Lambda_y + 0.5, \Lambda_z)\,,$$

in lattice coordinates, then transformed to the global coordinate system using equation (6.8), where

$$g = \left(\frac{\rho}{4}\right)^{-1/3}.$$

If anchorSpecifies = *corner* then the four positions above are shifted in the lattice coordinates by $(-0.25, -0.25, -0.25)$ prior to transformation.

6.4 Optimising the anchor position

The choice of anchor strongly influences the speed of the algorithm. An anchor near to the centre of the zone volume minimises the number of unnecessary unit cells to be tested. This has a large impact when the mesh has been decomposed for parallel processing because the anchor may lie far from the spatial region assigned to the processor in question. The algorithm will test many unnecessary unit cells before it reaches the region that the processor in question has been assigned, whereupon the expanding cube will be large and only a small proportion of the unit cells added to a layer will create molecules that are accepted.

The anchor position can also impact on the ability to fill disconnected volumes. If a volume of a zone finishes filling before another starts, then the algorithm will terminate and not fill the further away volume. Again, an anchor near the centre of the zone mitigates this. Both of these problems are addressed by automatically moving the anchor from the user-specified point to the lattice site that is closest to the zone volume centre; molecules are created at the same positions as they are when starting from the user-specified anchor, so no change in the configuration results.

The centroid of the volume is not the quantity of interest because this is weighted towards larger volumes. The required centre is instead calculated by finding the midpoint between the extremities of the volumes of the zone in the global coordinate system,

$$\mathbf{C} = \frac{1}{2}\left(X_{min} + X_{max}, Y_{min} + Y_{max}, Z_{min} + Z_{max}\right). \tag{6.9}$$

The closest lattice point to $\mathbf{C}$ is required, and will be called the optimised anchor, $\mathbf{A}_O$ (see figure 6.4). Rewriting equation (6.8) for $\mathbf{C}$, where $\mathbf{\Lambda}_C$ is the position of the volume centre in local lattice coordinates and $\mathbf{A}$ is the user specified

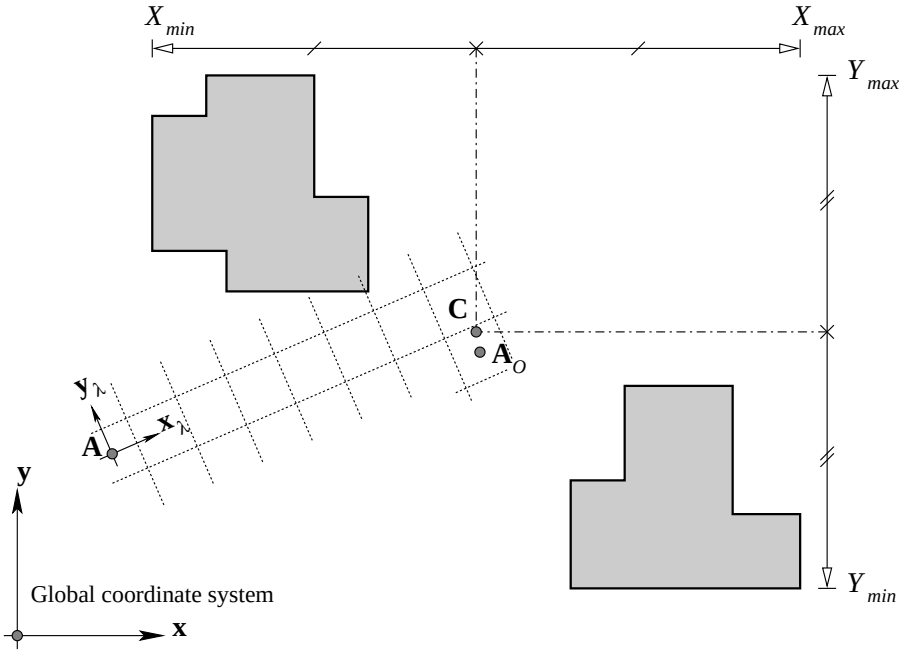


Figure 6.4: Placing the optimised anchor at the unit cell position closest to the centre of the volumes in the zone.

anchor,

$$\mathbf{C} = \mathbf{A} + \mathbf{R} \cdot (\mathbf{G} \cdot \mathbf{\Lambda}_C), \tag{6.10}$$

and rearranging for $\mathbf{\Lambda}_C$,

$$\mathbf{\Lambda}_C = \mathbf{G^{-1}} \cdot \left(\mathbf{R^{-1}} \cdot (\mathbf{C} - \mathbf{A})\right). \tag{6.11}$$

$\mathbf{R}$ is orthogonal, so $\mathbf{R^{-1}} = \mathbf{R^T}$ and

$$\mathbf{G^{-1}} = \begin{pmatrix} 1/g_{11} & 0 & 0 \\ 0 & 1/g_{22} & 0 \\ 0 & 0 & 1/g_{33} \end{pmatrix}. \tag{6.12}$$

The closest unit cell centre in lattice coordinates, $\mathbf{\Lambda}_{A_O}$, is found by

$$\mathbf{\Lambda}_{A_O} = (nint(\Lambda_{Cx}), nint(\Lambda_{Cy}), nint(\Lambda_{Cz})), \tag{6.13}$$

where *nint* returns the nearest integer to the argument. The optimised anchor is given finally by

$$\mathbf{A}_O = \mathbf{A} + \mathbf{R} \cdot (\mathbf{G} \cdot \mathbf{\Lambda}_{A_O}), \tag{6.14}$$

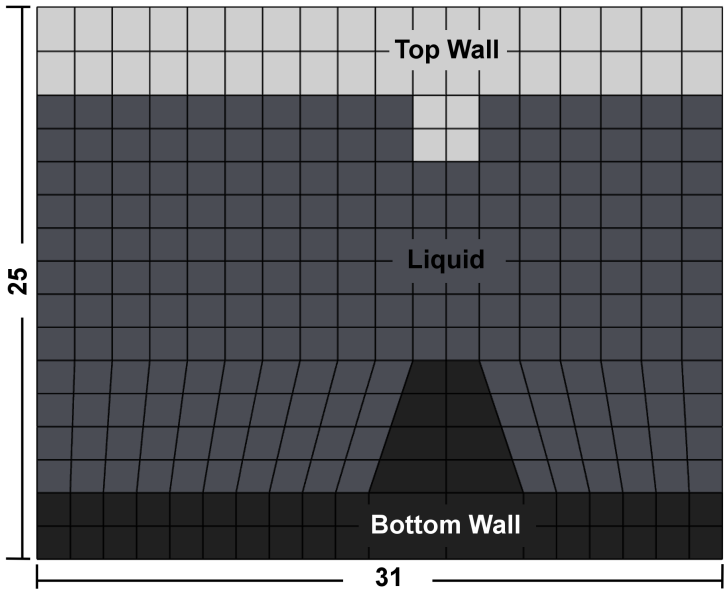


Figure 6.5: Three zone test case mesh and zones: a channel with a constriction. Overall dimensions are $31 \times 25 \times 12$ MD reduced units.

and $\mathbf{A}_O$ is used instead of $\mathbf{A}$ in equation (6.8) when generating unit cells.

Optimising the anchor position does not guarantee that all possible arrangements of disconnected volumes will be filled, although it substantially increases the number that will be. If a volume in a particular zone does not get filled with molecules, then the zone can be subdivided and the new zones given identical details in molConfigDict. The mesh can be redecomposed to distribute the cells amongst processors differently if the problem occurs only in parallel.

6.5 A three zone test case

The operation of the placement algorithm is illustrated by the example case illustrated in figure 6.5. Table 6.2 shows the pertinent molConfigDict entries describing the configuration of the system. Three different types of molecule are created, in three different lattice structures. The wall zones tether their molecules in place and the liquid region is given an initial average velocity. All numerical values are expressed in MD reduced units [130, 131] with an energy scale of $120k_b$, where k_b is the Boltzmann constant, a length scale of $0.34nm$ and a mass scale of $6.6904 \times 10^{-26}kg$.

Figure 6.6 shows the configuration of molecules that molConfig generates for

Table 6.2: Pertinent molConfigDict entries for the three zone test case in figure 6.5.

Entry	Bottom Wall	Liquid	Top Wall
density	0.8	0.8	0.85
bulkVelocity	(0.0 0.0 0.0)	(2.0 0.0 0.0)	(0.0 0.0 0.0)
id	WALLB	LJ	WALLT
latticeStructure	FCC	SC	BCC
anchor	(16.25 4.75 6.25)	(6.25 13.25 6.25)	(18.25 20.75 6.25)
anchorSpecifies	molecule	molecule	corner
tethered	yes	no	yes
orientationAngles	(30 0 0)	(45 0 0)	(0 0 0)

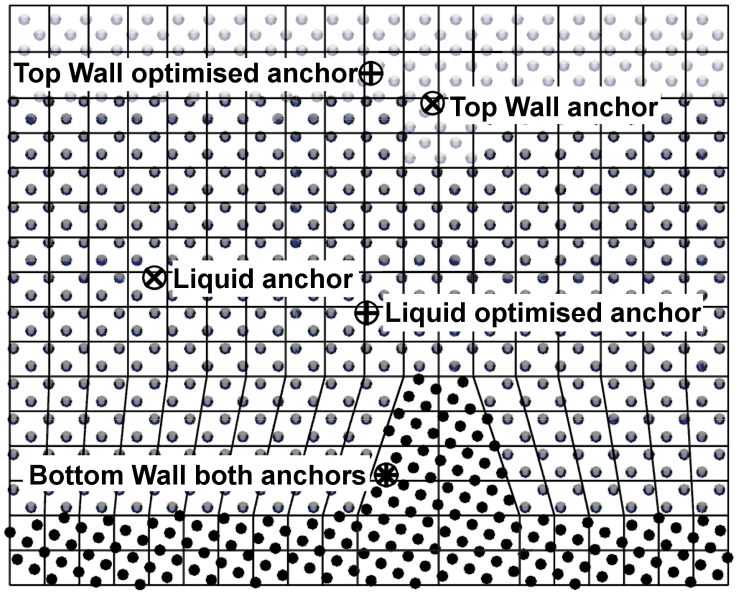


Figure 6.6: Three zone test case configuration of molecules. Molecules coloured according to id. User specified anchors ⊗ and optimised anchors ⊕ are shown.

this test case, and the positions of the user-specified and optimised anchors. The bottom wall anchor is in the optimal position. The bottom wall and liquid anchors are positioned on top of molecules and the top wall anchors are between molecules, as per their respective anchorSpecifies entries. Figure 6.7 shows the molecules in 3D.

The mesh is decomposed into four portions for parallel processing using the METIS [141] library (see figure 6.8). The mesh portions assigned to processors 1 and 3 contain disconnected volumes belonging to the same zone. Focusing on the portion assigned to processor 1 (see figure 6.9) there are two disconnected volumes each for the bottom wall and liquid zones. The optimised anchors for

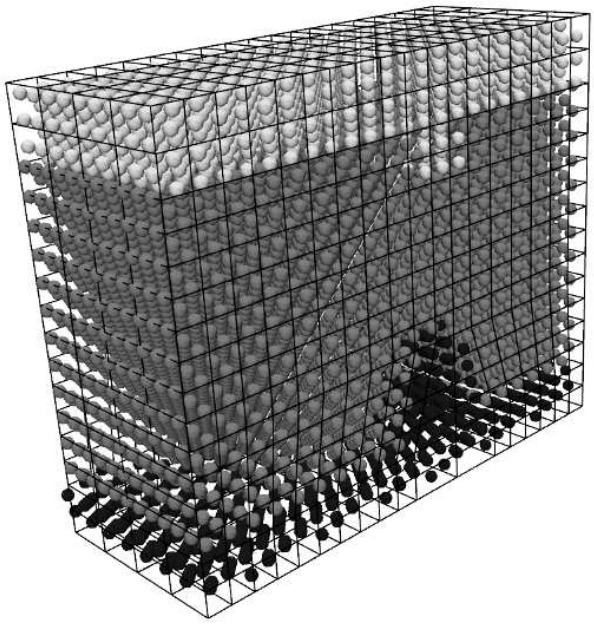


Figure 6.7: 3D view of the molecules created in the three zone test case. Molecules coloured according to id.

these two zones are shown. The same configuration of molecules is generated when the four portions are filled independently by different processors, as when the whole mesh is filled using one processor. Figures 6.5 to 6.9 were generated using ParaView [144].

6.6 Molecular velocity generation

The velocities of the molecules created are initialised to a set of thermal (random) velocities to match the user defined temperature, T_U, in the molConfigDict entry for the zone. The equilibrium kinetic temperature, T, of a stationary system of N_M molecules [131] is given by

$$T = \frac{1}{3N_M} \sum_{i=1}^{N_M} m_i v_i^2, \tag{6.15}$$

in MD reduced units, where m_i is the mass of a molecule and v_i is the magnitude of its velocity. The direction of the velocity assigned to each molecule is random, and the magnitude of each is chosen to give the correct temperature. The distribution of molecular speeds can either be Maxwellian or uniform.

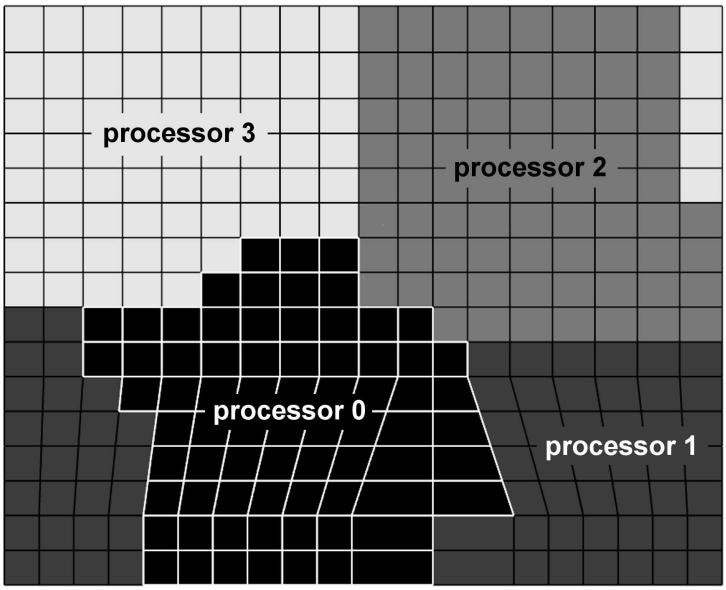


Figure 6.8: The mesh for the three zone test case decomposed into 4 portions for parallel processing. Processors 1 and 3 have zones with disconnected volumes.

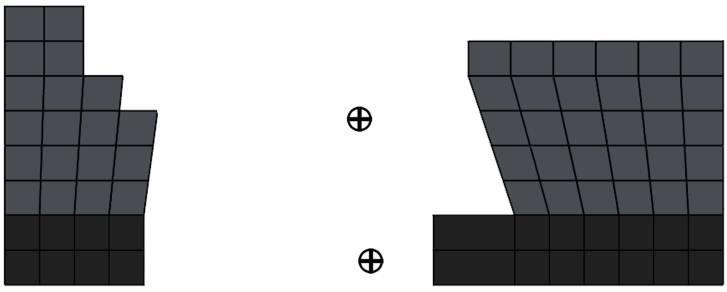


Figure 6.9: The disconnected volumes of the bottom wall and liquid zones residing on processor 1. The shading corresponds to the zones as shown in figure 6.5. The optimised anchors for filling each zone on this processor are shown.

Maxwellian distribution

The equilibrium (Maxwellian) velocity distribution of a system is defined as the probability of a component of a molecule's velocity in any direction being a normal distribution with a zero mean [145], i.e.

$$P(v_x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-v_x^2/2\sigma^2}, \tag{6.16}$$

where $\sigma = \sqrt{T/m}$. The velocity to assign to a molecule, $\mathbf{v}_i$, is obtained by sampling the component in each direction from a normal distribution random number generator with zero mean and variance $\sigma^2 = T_U/m_i$. A Maxwellian initial distribution creates molecular velocities that are physically more realistic than a uniform distribution.

Uniform distribution

A vector, $\mathbf{v}_r$, is created for each molecule which has each of its components generated by a uniform random number generator returning a value between -1 and 1. The velocity assigned to the molecule, $\mathbf{v}_i$, is given by

$$\mathbf{v}_i = \sqrt{\frac{3T_U}{m_i}} \frac{\mathbf{v}_r}{|\mathbf{v}_r|}. \tag{6.17}$$

The prefactor is obtained by rearranging equation (6.15) for v_i with $N_M = 1$. A uniform initial velocity distribution can be used to determine when a system has equilibrated: measuring the velocity distribution in the system as the simulation progresses and identifying when it reaches a Maxwellian form.

Remove drift velocity and apply average

The finite number of random numbers used to generate the molecular velocities will in general not result in a mean velocity of zero. This remaining 'drift' velocity is removed and the user-specified average velocity for the zone in question, $\mathbf{v}_U$, is added to each molecule by

$$\mathbf{v}_i' = \mathbf{v}_i - \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i + \mathbf{v}_U. \tag{6.18}$$

6.7 Avoiding high energy overlaps at zone boundaries

At the interface between different zones it is possible that molecules will be placed so close to each other that the high-energy, repulsive portions of their intermolecular potentials will overlap. This causes the MD simulation to crash because these high energy overlaps accelerate molecules to very large velocities in the first timestep, where they then overlap with the first molecule that they collide with (due to the finite timestep), accelerating it to a very high velocity. This causes a cascade of high energy impacts and the total energy in the simulation to increase uncontrollably. This is similar to what happens when too long an integration timestep is used, see section 8.2.1.

This problem is avoided by calculating the intermolecular energy between pairs of molecules (see chapter 7) before the first timestep; any pairs sharing a potential energy above a user defined threshold, U_{max}, are identified. If the molecules in a high energy pair have the same id (are the same type of molecule) then one of them is deleted from the simulation. If the molecules have different ids then a user specified removal order list is consulted. The molecule that appears first in the removal order list is deleted. An example of this is shown in figure 6.10. This also applies across periodic and interprocessor boundaries, where molecules on either side of these boundaries can be so close to each other that they overlap.

Determining a potential energy limit

There are numerous ways to choose a potential energy limit. Each of them has the same objective — identify a pair-interaction energy that, if converted to kinetic energy, would impart a velocity to the molecules in the pair that would cause an energy cascade for the current value of timestep. Currently a single energy limit is used that must cope with the 'worst case' overlap. It would be possible to have a different limit corresponding to each intermolecular potential.

A simple method of determining U_{max} is to choose a velocity from the high speed end of the Maxwellian velocity distribution for the 'worst-case' situation — the lightest molecules at the highest temperature. The velocity chosen should be relatively improbable — see appendix C for how to choose a velocity with a specified probability. For example, for a temperature of $T = 2.5$ and molecules of mass $m = 1$ (all quantities in MD reduced units), then there is a probability

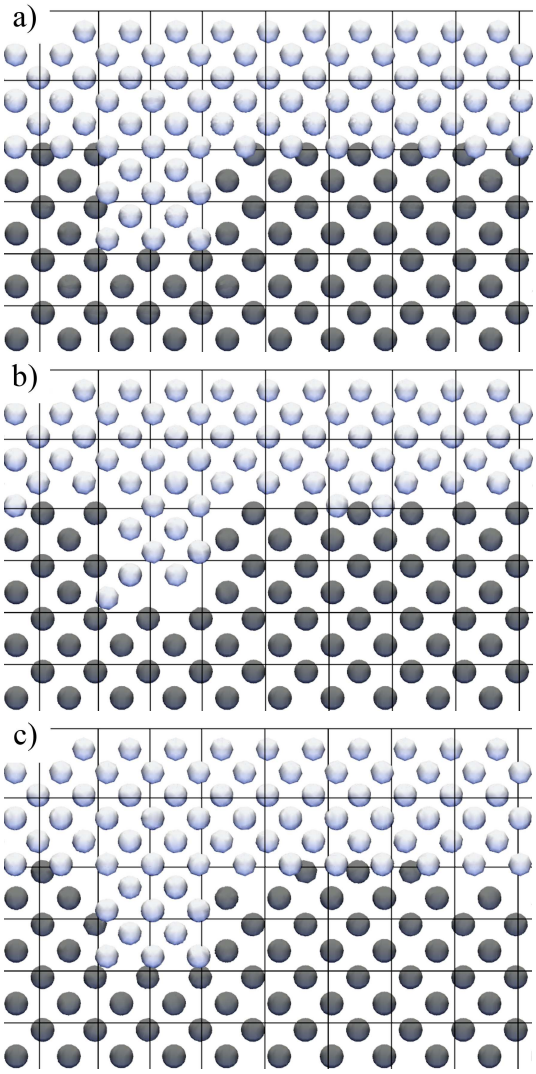


Figure 6.10: A close-up of the top right of figure 6.6 showing the constriction in the top wall zone. a) shows the molecules created by filling the zones, note the overlapping molecules at the interface. b) the result when WALLT comes before LJ (see table 6.2) in the removal order list: liquid zone molecules (LJ) are retained at the expense of top wall molecules (WALLT). c) the result when LJ comes before WALLT in the removal order list: top wall molecules (WALLT) are retained at the expense of liquid zone molecules (LJ).

of 1/1000 that a particular molecule will be travelling at a speed of $v = 6.756$. Therefore,

$$U_{max} = \frac{1}{2}mv^2 = \frac{6.756^2}{2} = 22.82,$$

would be a suitable potential energy limit in this case. A limit should be low enough to ensure stability, but not so low as to remove large numbers of molecules that are legitimately interacting, should the simulation be stopped and restarted at any point.

Insertion of complex molecules

The high energy overlap removal mechanism can be used as the basis for the insertion of complex molecules. For example, the insertion of large biomolecules into a system, or adding carbon nanotubes spanning a solid barrier between two fluid reservoirs. The complex molecule can be created in place and, providing that the solid and solvent molecules appear before any of the constituents of the complex molecule in the removal order list, it will be embedded into the existing matter.

Chapter 7

Intermolecular Force Calculation

7.1 Interacting cell identification

In order to efficiently calculate intermolecular pair forces in meshes constructed from unstructured arbitrary polyhedral cells, the Arbitrary Interacting Cells Algorithm (AICA) was devised. AICA is a generalisation of the Conventional Cells Algorithm (CCA) [130, 131]. In the CCA, a simple (usually cuboid) simulation domain is subdivided into equally sized cells. The minimum dimension of the CCA cells must be greater than r_{cut}, the cut-off radius of the intermolecular pair potential, so that all molecules in a particular cell interact with all other molecules in their own cell and with those in their nearest neighbour cells (i.e. those they share a face, edge or vertex with — 26 in 3D). Extensions to this that permit the use of smaller cells [146–148] have been proposed, but they still require the domain to comprise a structured mesh of uniform cubes or cuboids. The objective of any cell algorithm is to evaluate interactions between as few molecules as possible that are further apart than r_{cut} to maximise computational speed.

AICA uses a 3D mesh of unstructured polyhedra so there are no restrictions on cell size, shape or connectivity. A cell's position in the mesh structure does not imply anything about its physical location or connectivity. Therefore, the ability to handle arbitrary geometries comes from deducing locally which cells are in range to interact. Each cell has a unique list of other cells that it is to interact with, this list is known as the Direct Interaction List (DIL) for the Cell In Question (CIQ). It is constructed by searching the mesh to create a set of cells that have at least one part of their surface within a distance of r_{cut} from the surface of the CIQ, see figure 7.1. Where there are multiple molecular species

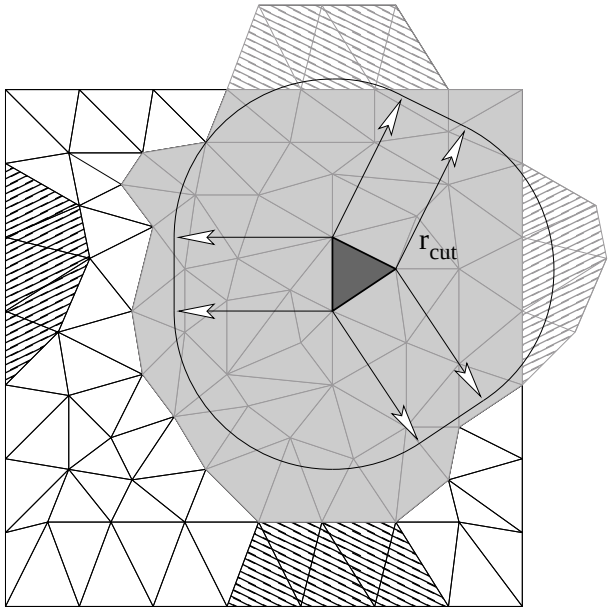


Figure 7.1: Interacting cell identification example using 2D unstructured triangular cells. The spatial domain (main square section comprising the real cells) is periodic top-bottom and left-right. Real cells within r_{cut} that interact with the CIQ (dark) are shaded in grey. The required referred cells are hatched in alternate directions according to which boundary they have been referred across. Realistic systems would be significantly larger compared to r_{cut} than shown here.

present with different values of r_{cut}, the maximum value is used to determine cell interactions. Where substantially different cut-off radii are present, it would be possible to have separate DILs for different range interactions, all of which could be constructed simultaneously during a single evaluation of the mesh.

This work has been carried out on the basis that the mesh is static. It is possible to use the same methods for a system with a dynamic mesh, providing that the information about which cells interact can be efficiently and reliably kept up-to-date, or rebuilt without incurring an unreasonable computational cost.

The DILs are established prior to the start of simulation and are valid throughout because the spatial relationship of the cells is fixed, whereas the set of molecules they contain is dynamic. In a similar way to the CCA, at every timestep a molecule in a particular cell calculates its interactions with the other molecules in that cell and consults the cell's DIL to find which other cells contain molecules it should interact with. Information is required to be maintained at all times stating which cell a molecule is in — this has been implemented in a robust and efficient manner in OpenFOAM as part of the method for tracking particle motion, see section 8.3.

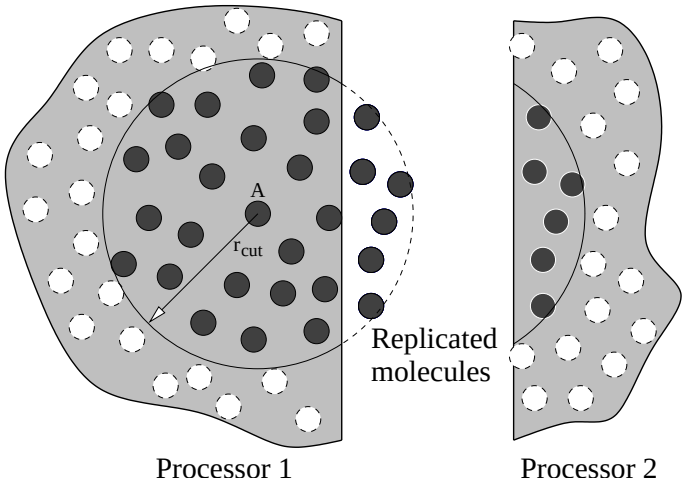


Figure 7.2: Spatial decomposition for parallel processing. Molecule A must calculate intermolecular forces with all other molecules within its cut-off radius, r_{cut}. When the domain has been decomposed, some of these molecules may lie on a different processor. In this case, copies of the appropriate molecules from processor 2 are made on processor 1.

7.2 Replicated molecule periodicity and parallelisation

When parallelising an MD simulation, the spatial domain is decomposed and each processor is given responsibility for a single region. Molecules that cross the boundaries between these regions need to be communicated from one processor to the next. Processors also communicate when carrying out intermolecular force calculations, in which molecules close to processor boundaries need to be replicated on their neighbours to provide interactions. This process is illustrated in figure 7.2. Periodic boundaries also require information about molecules that are not physically adjacent in the domain (see figure 7.3); these required interactions can also be constructed by creating copies of molecules outside the boundary.

It is possible to handle processor and periodic boundaries in exactly the same way, because they have the same underlying objective: molecules near to the edge of a region need to be copied either between processors or to other locations on the same processor at every timestep to provide interactions. This is a useful feature because decomposing a mesh for parallel processing will often turn a periodic boundary into a processor boundary. The issue is: how to efficiently identify which molecules need to be copied, and to which location, because this set continually changes as the molecules move.

More general and flexible coupled boundaries can be implemented using the

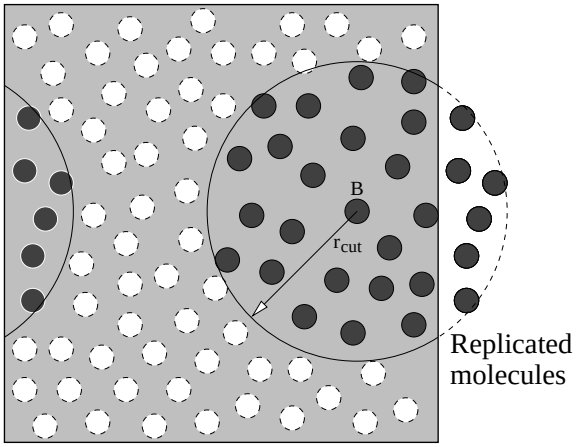


Figure 7.3: Periodic boundaries. Molecule B must calculate intermolecular forces with all other molecules within r_{cut}. Some of those molecules may be on a periodic image of the system, which, in computational terms, will reside on the other side of the domain. Serial calculations in simple geometries typically use the minimum image convention [130, 131], but this is not suitable for parallelisation.

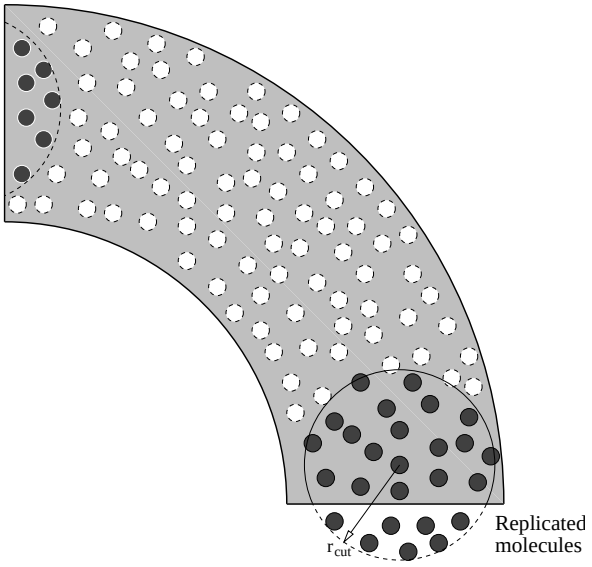


Figure 7.4: An example of a non-parallel, separated boundary. This could represent either a 90° bend in a channel where molecules are 'recycled' from inlet to outlet. It could also represent a cross section through a hybrid simulation of flow in a pipe where the MD section is an outer annulus and rotational symmetry has been exploited to simulate only a quarter of the pipe — a typical CFD technique. In both cases, molecules near the separated boundaries must correctly exchange intermolecular forces.

same framework, where molecules are replicated to either side of a spatially-separated, non-parallel boundary. For example, a 'recycling' boundary condition or a rotationally symmetric simulation, see figure 7.4, or coupling an inlet or outlet boundary or hybrid interface with any orientation to a static, external molecule reservoir.

7.3 Referred molecules and cells

Replicated molecule parallelisation and periodic boundaries are handled in the same way using *referred cells* (see figure 7.1) and *referred molecules*.

7.3.1 Referred molecule

A referred molecule is a copy of a real molecule that has been placed in a region outside a periodic or processor boundary in order to provide the correct intermolecular interaction with molecules inside the domain. A referred molecule holds only its own position and id (identification of which type of molecule it is for multi-species simulations). Referred molecules are created and discarded at each timestep, and do not report any information back to their source molecules. Therefore if molecule j on processor 1 needs to interact with molecule k on processor 2, a separate referred molecule will be created on each processor.

7.3.2 Referred cell

Referred cells define a region of space and hold a collection of referred molecules. Each referred cell knows

- which real cell in the mesh (on which processor) is its source;
- the required transformation to refer the position and orientation of the real molecules in the source cell to the referred location, see appendix E. A general transformation of position and orientation is required for the cases mentioned in section 7.2 and figure 7.4 which require the replicated cells to have a different orientation to their source cell;
- the positions of all of its own vertices. These are the positions of the vertices of the source cell which have been transformed by the same referring transform as the referred molecules it contains;

- the structure of all of the cell edges. Each edge is defined by a pair of indices in the referred vertex positions list indicating the vertices that comprise it;
- the structure of all of the cell faces. Each face is defined by a list of indices in the referred vertex positions list indicating the vertices that comprise it;
- which real cells are in range of this particular referred cell and hence require intermolecular interactions to be calculated. This is constructed once at the start of the simulation, in the same way as the DIL for real-real cell interactions.

7.4 Interacting cell identification methods

Building DILs and creating referred cells depends on identifying whether or not two cells are close enough such that the molecules they contain need to interact. Four methods for testing this were considered: PP, PPGR, CGAL and PFEE.

Point-Point (PP)

The fastest and most straightforward method of determining whether cells are within r_{cut} is to use Point-Point (PP) searching. All of the N_p points in the mesh are compared to each other using a non-double-counting loop, and if any pair of points are within r_{cut} of each other, then all of the cells that these points form part of[1] must all be in range of each other, see algorithm 1. Note that a point is compared to itself ($p_j = p_i$ is used for the inner loop rather than $p_j = p_i + 1$) — this guarantees that neighbouring cells are identified for interaction if none of their non-shared vertices are within range of each other.

It is possible for PP to introduce errors, where molecules that should interact do not because their containing cells are not identified as being in range. Slices of cells may not be identified for interaction because

- a vertex may be within r_{cut} of a face of another cell, but not of any vertex. See figure 7.5;

[1] The information about which cells use a specific mesh point, and all subsequent information about the relationship between the points, edges, faces and cells in the mesh exists in, and is readily accessible from, the mesh description in OpenFOAM.

Algorithm 1 Point-Point cell searching

```
for p_i = 0; p_i < N_p; p_i + 1 do
  for p_j = p_i; p_j < N_p; p_j + 1 do
    if the magnitude of the separation of the points referenced by p_i and
    p_j ≤ r_cut then
      all cells which p_j and p_i form part of must interact.
    end if
  end for
end for
```

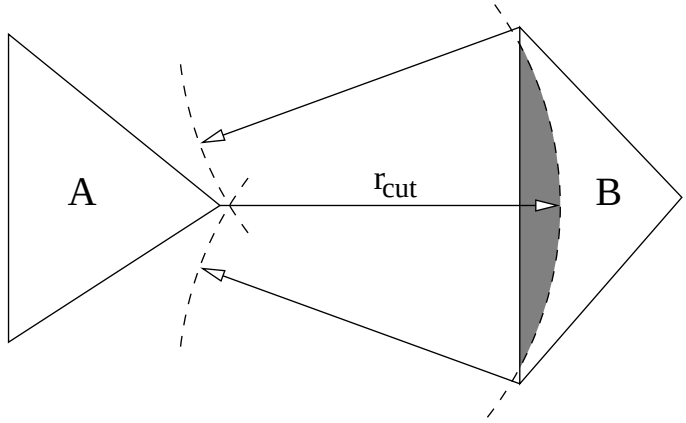


Figure 7.5: Errors caused by cut-off radius searching between vertices only. A sphere of radius r_{cut} drawn from the indicated vertex on cell A intersects a face of cell B, therefore, molecules near this vertex should interact with molecules in the shaded region. A sphere of radius r_{cut} drawn from either of the indicated vertices on cell B will not intersect any point of the surface of cell A; therefore, point-face searches are not reciprocal.

- the edges of two cells may be within r_{cut} of each other, but none of the vertices of either cell are within r_{cut} of a vertex, edge or face of the other. See figure 7.6, this problem is only possible in 3D.

Both of these problems are most prevalent in meshes of tetrahedral cells, and cannot occur in regular meshes with cuboid, or nearly cuboid cells.

Point-Point with Guard Radius (PPGR)

The errors identified with the PP method can be reduced by adding a guard radius, r_G, to r_{cut}. PPGR works exactly as PP, except cells with vertices separated by $\leq r_G + r_{cut}$ now also interact. The guard radius will mean that many cells will have DILs that are larger than necessary, which will slow down the running of the simulation because more unnecessary molecule pairs will be evaluated at each timestep. It is possible to remove all errors from PP by adding a large enough

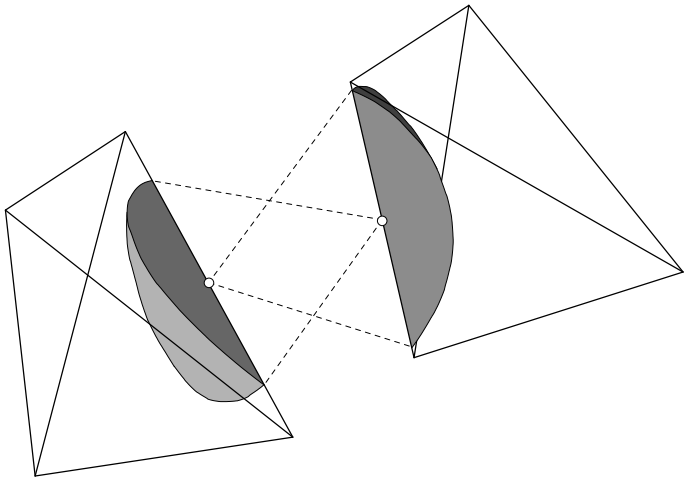


Figure 7.6: Two tetrahedral cells with edges within r_{cut} (dashed lines) of each other, but searching from vertices cannot establish this. The closest points between the two edges are marked, and the intersecting volumes on the other cell of a sphere of radius r_{cut} drawn from these points is shaded. Note that the straight edges of the shaded sections are perpendicular.

guard radius, but this value is difficult to determine in advance, and in practise is relatively large, introducing a significant running performance penalty.

Computational Geometry Algorithms Library (CGAL)

Whether cells need to interact can be found without either the errors of PP or the unnecessary additions to DILs of PPGR. The published CGAL library [99] can be used to calculate the closest distance between two convex hulls [149], constructed from the points of each cell. This method always finds all of the cells that need to interact, irrespective of their relative geometry, by using a single non-double-counting loop comparing all cells in the mesh to each other. When implemented, however, the computational cost proved *thousands* of times greater than any of the other three methods described here. It is not suitable for meshes with a realistic number of cells, and will not be discussed further.

Point-Face and Edge-Edge (PFEE)

All cells within r_{cut} of each other can be identified, without adding any unnecessary cells to a DIL, by conducting the mesh search on the basis of the two errors identified in PP: point-face and edge-edge searching.

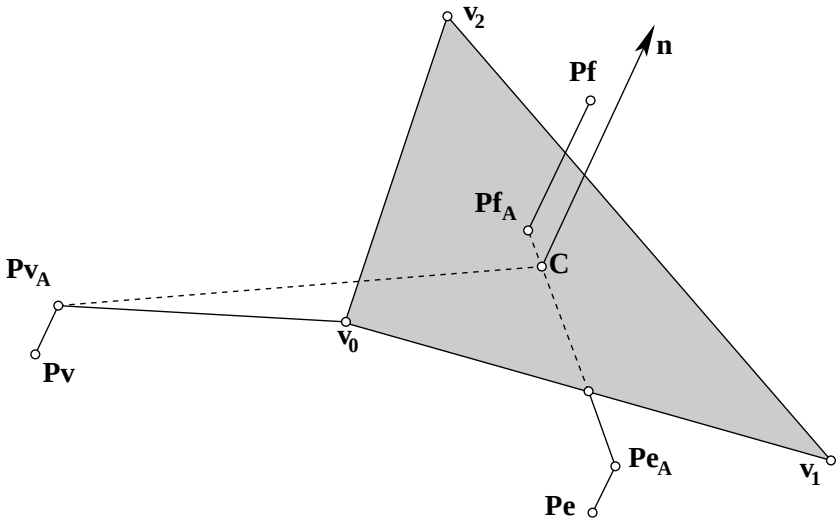


Figure 7.7: The three possibilities for the closest point on a face to an arbitrary point; the point can be closest to a vertex, **Pv**, an edge, **Pe**, or the face itself, **Pf**. Points labelled with a subscript **A** are those projected onto the plane of the face.

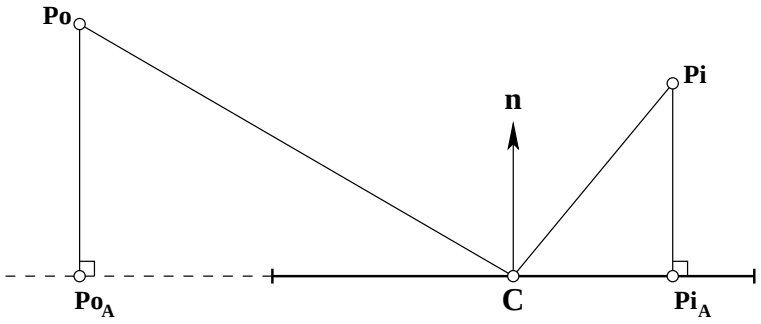


Figure 7.8: Projecting the point to be evaluated onto the plane defined by the face centre and normal unit vector. The projected point can either lie outside ($\mathbf{Po_A}$) or inside ($\mathbf{Pi_A}$) the face. The face is shown as a solid line and the extended plane as a dashed line.

Point-Face The closest point on a face (which is a polygon with an arbitrary number of sides) to an arbitrary 3D point, $\mathbf{P}$, can either be a face vertex, lie on an edge, or lie on the face itself. Figure 7.7 shows these three cases. The question of whether the point is within r_{cut} of the face can be determined by the following algorithm. The structure of this point-face algorithm is such that it only evaluates as much of the problem as is necessary to make a decision about whether the point is in range of the face. Once it has this, it stops the evaluation of the current point-face pair and moves on to the next.

1. Project $\mathbf{P}$ to the nearest point, $\mathbf{P_A}$, on the plane defined by the face centre, $\mathbf{C}$ and normal unit vector, $\mathbf{n}$, see figure 7.8, using

$$\mathbf{P_A} = \mathbf{P} - ((\mathbf{P} - \mathbf{C}) \cdot \mathbf{n})\,\mathbf{n}. \tag{7.1}$$

   If $|(\mathbf{P} - \mathbf{C}) \cdot \mathbf{n}| > r_{cut}$ then it is not possible for the point to be in range of the face[2]. No more calculation is necessary for this point-face pair.

2. Whether $\mathbf{P_A}$ lies inside or outside of the face must be determined. A line $\mathbf{P_AC}$ is drawn from $\mathbf{P_A}$ to $\mathbf{C}$, along the plane of the face and *each edge of the face is tested to see if this line crosses it.* To do this, all of the vertices, $\mathbf{v_0}, \mathbf{v_1} \ldots \mathbf{v_N}$, are projected onto a local 2D coordinate system that coincides with the plane of the face, with $\mathbf{P_A}$ as its origin, see figure 7.9. This is necessary because $\mathbf{P_AC}$ and an edge may be skew if considered as 3D lines due to numerical round-off errors in the position of points of the face, or where a face is not perfectly flat, which is possible when more than three vertices are used to define it[3]. The axis unit vectors of the coordinate system on the plane are

$$\mathbf{x}' = \frac{\mathbf{C} - \mathbf{P_A}}{|\mathbf{C} - \mathbf{P_A}|}, \tag{7.2}$$

$$\mathbf{y}' = \frac{(\mathbf{C} - \mathbf{P_A}) \times \mathbf{n}}{|(\mathbf{C} - \mathbf{P_A}) \times \mathbf{n}|}, \tag{7.3}$$

[2]The test actually performed is $|(\mathbf{P} - \mathbf{C}) \cdot \mathbf{n}|^2 > r_{cut}^2$ to avoid the square root when determining the magnitude of the vector. All other vector magnitude comparisons are also performed in this way.

[3]The presence of non-planar faces in the mesh does not cause any errors in point-face searching because which cell a molecule occupies in the mesh is also determined by the effective plane of the face defined by the face centre and normal vector(see section 8.3.

and the position vector of a vertex, $\mathbf{v}_\alpha$, in plane coordinates is

$$\mathbf{v'}_\alpha = ((\mathbf{v}_\alpha - \mathbf{P_A}) \cdot \mathbf{x'})\,\mathbf{x'} + ((\mathbf{v}_\alpha - \mathbf{P_A}) \cdot \mathbf{y'})\,\mathbf{y'}. \tag{7.4}$$

Given that $\mathbf{C}$ lies on the line of $\mathbf{x'}$, by definition

$$\mathbf{C'} = |\mathbf{C} - \mathbf{P_A}|\,\mathbf{x'}. \tag{7.5}$$

When determining the local coordinates of the vertices, if $|\mathbf{P} - \mathbf{v}_\alpha| \leq r_{cut}$ then the face must be in range of the point. No more calculation is necessary for this point-face pair.

3. Finding $\mathbf{E}$, the intersection point between $\mathbf{P_A C}$ and the edge defined by vertices $\mathbf{v}_\alpha$ and $\mathbf{v}_\beta$. The equations for $\mathbf{E'}$, the intersection in plane coordinates of the lines $\mathbf{P'_A C'}$ and $\mathbf{v'}_\alpha\mathbf{v'}_\beta$, are

$$\mathbf{E'} = \lambda_A \mathbf{C'}, \tag{7.6}$$
$$\mathbf{E'} = \mathbf{v'}_\alpha + \lambda_v \left(\mathbf{v'}_\beta - \mathbf{v'}_\alpha\right). \tag{7.7}$$

Expanding equations (7.6) and (7.7) in $\mathbf{x'}$ and $\mathbf{y'}$ components

$$\begin{aligned}
E'_{x'} &= \lambda_A C'_{x'}, \\
E'_{y'} &= \lambda_A C'_{y'}, \\
E'_{x'} &= v'_{\alpha x'} + \lambda_v \left(v'_{\beta x'} - v'_{\alpha x'}\right), \\
E'_{y'} &= v'_{\alpha y'} + \lambda_v \left(v'_{\beta y'} - v'_{\alpha y'}\right),
\end{aligned}$$

and solving for λ_A and λ_v, given that $C'_{y'} = 0$ because $\mathbf{E'}$ lies on the line of $\mathbf{x'}$

$$\lambda_A = \frac{1}{C'_{x'}} \left(v'_{\alpha x'} - v'_{\alpha y'} \frac{\left(v'_{\beta x'} - v'_{\alpha x'}\right)}{\left(v'_{\beta y'} - v'_{\alpha y'}\right)} \right), \tag{7.8}$$

$$\lambda_v = -\frac{v'_{\alpha y'}}{v'_{\beta y'} - v'_{\alpha y'}}. \tag{7.9}$$

If $0 \leq \lambda_A \leq 1$ and $0 \leq \lambda_v \leq 1$, then the edge in question is crossed between $\mathbf{P_A}$ and $\mathbf{C}$. $\mathbf{P_A}$ must have been outside of the face, and $\mathbf{E}$ is the closest

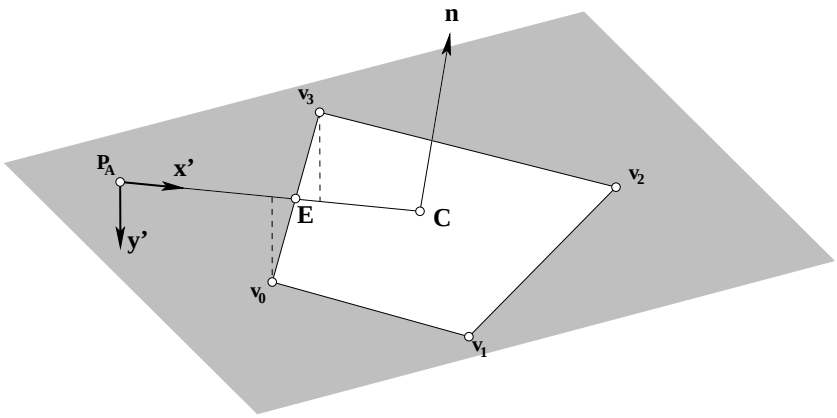


Figure 7.9: Defining a coordinate system local to the plane of the face to calculate intersections with face edges.

point on the face to $\mathbf{P}$,

$$\mathbf{E} = \mathbf{P_A} + \lambda_A \left(\mathbf{C} - \mathbf{P_A} \right). \tag{7.10}$$

If $|\mathbf{P} - \mathbf{E}| \leq r_{cut}$ then the face is in range of the point. If no edge is crossed by $\mathbf{P_A C}$, then $\mathbf{P_A}$ must have been on the face, so if $|\mathbf{P} - \mathbf{P_A}| \leq r_{cut}$ then the face is in range of the point, and the cells that the face forms part of must interact with the cells that the point forms part of.

The non-reciprocal nature of the point-face search must be borne in mind when searching for cells using it. When constructing DILs all faces must be tested with all points; no reduction in cost via a non-double-counting procedure is possible. Real cell points must search for referred cell faces and referred cell points must search for real cell faces. This, and the relatively complex algorithm for calculating the point-face distance, results in PFEE being computationally expensive.

Edge-Edge This uses the derivation of the closest distance between two skew lines [150] to determine whether two edges are within range of each other. The edges to be compared (see figure 7.10) lie on lines that extend to infinity, the equations for the closest points on these lines are

$$\mathbf{c_1} = \mathbf{v}_\alpha + \lambda_1 \left(\mathbf{v}_\beta - \mathbf{v}_\alpha \right), \tag{7.11}$$

$$\mathbf{c_2} = \mathbf{v}_\gamma + \lambda_2 \left(\mathbf{v}_\delta - \mathbf{v}_\gamma \right). \tag{7.12}$$

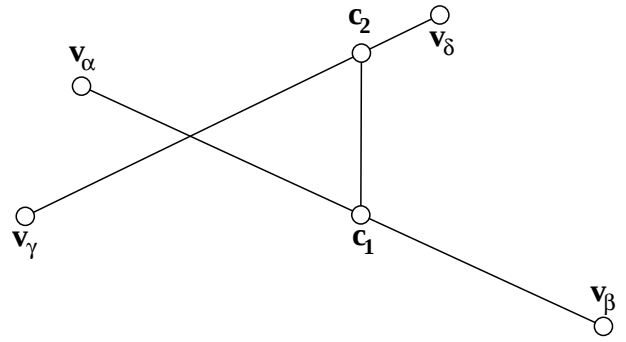


Figure 7.10: The closest distance between two edges.

Defining [151],

$$\mathbf{a} = \mathbf{v}_\beta - \mathbf{v}_\alpha,$$
$$\mathbf{b} = \mathbf{v}_\delta - \mathbf{v}_\gamma,$$
$$\mathbf{c} = \mathbf{v}_\gamma - \mathbf{v}_\alpha,$$

and solving equations (7.11) and (7.12) for λ_1 and λ_2 gives

$$\lambda_1 = \frac{(\mathbf{c} \times \mathbf{b}) \cdot (\mathbf{a} \times \mathbf{b})}{|\mathbf{a} \times \mathbf{b}|^2}, \tag{7.13}$$

$$\lambda_2 = \frac{(\mathbf{c} \times \mathbf{a}) \cdot (\mathbf{a} \times \mathbf{b})}{|\mathbf{a} \times \mathbf{b}|^2}. \tag{7.14}$$

If $|\mathbf{a} \times \mathbf{b}| = 0$, then the lines are parallel and this algorithm is invalid. A point-face search will, however, identify parallel edges as being in range if necessary. No more calculation is necessary for this edge pair.

If $0 \leq \lambda_1 \leq 1$ and $0 \leq \lambda_2 \leq 1$, then the closest points between the lines lie somewhere along both edges. In this case, calculate $\mathbf{c_1}$ and $\mathbf{c_2}$ using equations (7.11) and (7.12) and test $|\mathbf{c_1} - \mathbf{c_2}| \leq r_{cut}$ to determine if the edges, hence the cells they form part of, are in range. Edge-edge searching is reciprocal and can be performed under a non-double-counting loop.

7.5 Intermolecular force calculation procedure

Build cell interactions

At the start of the simulation, the DIL for each real cell and appropriate referred cells are created, and the referred cells determine which real cells they must supply interactions to. Details of these processes can be found in appendix D. Referred cells need not be sourced only from processors sharing a boundary, i.e. AICA is able to identify cell interactions between non-neighbouring processors. However, when constructing the referred cells, processors need only communicate across interprocessor faces, i.e. with neighbours only.

Build cell occupancy and refer molecules

At each timestep, before calculating any forces, a list for each cell is built stating which molecules it contains. This is a computationally cheap operation because it only involves querying each molecule for which cell it is in, and adding a reference or pointer to that molecule to the list for the appropriate cell. Each molecule holds information about which cell it occupies by virtue of the tracking mechanism (see section 8.3).

At each timestep, all real cells which are the source cell of one or more referred cells send the position and id of all of the molecules they contain to the appropriate referred cell on the appropriate processor. The destination referred cells perform the appropriate position and orientation transformation when the molecules are received. The referred molecules are discarded and re-created from the source molecules at each timestep.

Force calculation

The total intermolecular force acting on a molecule is calculated at each timestep by considering two types of interactions. **Real-Real** interactions occur between a molecule and others on the same processor, according to algorithm 2. **Real-Referred** interactions occur between real molecules and referred molecules arising from the other side of a processor or periodic boundary, according to algorithm 3. Referred molecules do not need to calculate interactions between themselves, because every referred molecule is a copy of a real molecule elsewhere, and as such will receive all of its real-real and real-referred interactions in-situ.

Algorithm 2 Real-Real Molecule Force Calculation.

for all real cells, c_i **do**
  for all real molecules in c_i, m_i **do**
    for all real cells in c_i's DIL, c_j **do**
      for all real molecules in c_j, m_j **do**
        calculate intermolecular force $\mathbf{f}_{ij}$ between m_i and m_j
        add $\mathbf{f}_{ij}$ to $\mathbf{f}_i$ (total force vector for m_i)
        add $\mathbf{f}_{ji} = -\mathbf{f}_{ij}$ to $\mathbf{f}_j$
      end for
    end for
    for all real molecules in c_i with an index greater than* m_i, $m_{i'}$ **do**
      calculate intermolecular force $\mathbf{f}_{ii'}$ between m_i and $m_{i'}$
      add $\mathbf{f}_{ii'}$ to $\mathbf{f}_i$
      add $-\mathbf{f}_{ii'}$ to $\mathbf{f}_{i'}$
    end for
  end for
end for

*Comparison of the index of molecules in the same cell, $m_i' > m_i$, means that interactions between molecules in the same cell are not double-counted and a molecule does not calculate an interaction with itself. The order of comparison is not important, as long as all pairs are covered, so $m_i' < m_i$ would work equally well.

Algorithm 3 Real-Referred Molecule Force Calculation

for all referred cells, c_q **do**
  for all referred molecules in c_q, m_q **do**
    for all real cells that c_q interacts with, c_p **do**
      for all real molecules in c_p, m_p **do**
        calculate intermolecular force $\mathbf{f}_{pq}$ between m_p and m_q
        add $\mathbf{f}_{pq}$ to $\mathbf{f}_p$
      end for
    end for
  end for
end for

Chapter 8

Integration of equations of motion

8.1 Integration algorithms

The equation of motion for the rotationally symmetrical molecules used in this work is simply Newton's second law of motion. In general this is not the case, and the Lagrangian or Hamiltonian [143, 152] formulation must be used to derive the equations for motion of molecules with rotational and internal degrees of freedom. There are two classes of numerical integration schemes in wide-spread use in molecular dynamics: leapfrog and predictor-corrector. The leapfrog scheme is simple and gives better energy conservation for a given timestep. The predictor-corrector method is higher order, giving better trajectory accuracy, and is more flexible for complex problems; it is preferred when solving equations of motion involving the rotational dynamics of molecules and constraint dynamics [130]. Only leapfrog integration has been implemented so far, although the infrastructure has been established for efficient runtime selection of predictor-corrector integration, anticipating the widening of the complexity of problems that will be solved.

A detailed evaluation of what situation each is best applied in and their relative merits will not be included (for this see [130, 131]).

8.1.1 Leapfrog

For a molecule at a position $\mathbf{r}$ at time t, advancing by a timestep of Δt, forming a Taylor expansion of a directional component, r:

$$r(t+\Delta t) = r(t) + \frac{dr(t)}{dt}\Delta t + \frac{1}{2!}\frac{d^2r(t)}{dt^2}\Delta t^2 + \frac{1}{3!}\frac{d^3r(t)}{dt^3}\Delta t^3 + O(\Delta t^4). \quad (8.1)$$

Here the velocity component of the molecule is

$$v(t) = \frac{dr(t)}{dt},$$

and the acceleration component is

$$a(t) = \frac{d^2r(t)}{dt^2},$$

which is known from the total intermolecular force acting on the molecule and Newton's second law: $\mathbf{f} = m\mathbf{a}$. Forming the Taylor expansion for $r(t-\Delta t)$,

$$r(t-\Delta t) = r(t) - \frac{dr(t)}{dt}\Delta t + \frac{1}{2!}\frac{d^2r(t)}{dt^2}\Delta t^2 - \frac{1}{3!}\frac{d^3r(t)}{dt^3}\Delta t^3 + O(\Delta t^4), \quad (8.2)$$

adding equations (8.1) and (8.2) (the $O(\Delta t^3)$ terms cancel), then rearranging for $r(t+\Delta t)$:

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + a(t)\Delta t^2 + O(\Delta t^4). \quad (8.3)$$

Dividing equation (8.3) by Δt and rearranging

$$\frac{r(t+\Delta t) - r(t)}{\Delta t} = \frac{r(t) - r(t-\Delta t)}{\Delta t} + a(t)\Delta t + O(\Delta t^3), \quad (8.4)$$

then substituting in

$$v(t+\Delta t/2) = \frac{r(t+\Delta t) - r(t)}{\Delta t}, \quad (8.5)$$

and

$$v(t-\Delta t/2) = \frac{r(t) - r(t-\Delta t)}{\Delta t}, \quad (8.6)$$

to give

$$v(t+\Delta t/2) = v(t-\Delta t/2) + a(t)\Delta t + O(\Delta t^3). \quad (8.7)$$

Returning to equation (8.3) and rearranging the right hand side into

$$\begin{aligned} r(t+\Delta t) &= r(t) + \left(\frac{r(t) - r(t-\Delta t)}{\Delta t} + a(t)\Delta t + O(\Delta t^3) \right) \Delta t, \\ &= r(t) + \left(v(t - \Delta t/2) + a(t)\Delta t + O(\Delta t^3) \right) \Delta t, \\ &= r(t) + v(t + \Delta t/2)\Delta t + O(\Delta t^4). \end{aligned} \tag{8.8}$$

Equations (8.7) and (8.8) constitute the leapfrog scheme for updating the velocity and position of the molecules at each timestep. Note that the position and velocity are evaluated for different times. This does not pose a problem, and the velocity can be recorded at the same time as the position by splitting its increment in two [130]. The sequence of calculation is then (using full vector quantities rather than components):

$$\mathbf{v}(t + \Delta t/2) = \mathbf{v}(t) + \mathbf{a}(t)\frac{\Delta t}{2}, \tag{8.9}$$

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t + \Delta t/2)\Delta t, \tag{8.10}$$

calculate $\mathbf{a}(t + \Delta t)$ using the updated positions, $\mathbf{r}(t + \Delta t)$, then,

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t + \Delta t/2) + \mathbf{a}(t + \Delta t)\frac{\Delta t}{2}. \tag{8.11}$$

8.2 Choosing an integration timestep

The timestep to use for integrating the equations of motion requires a trade-off between accuracy and computational speed. Shorter timesteps give a more accurate solution of the equation of motion and better conservation of momentum and energy. However, for the same simulation duration, more timesteps are required, hence more computational time. Experimentation is required with a small system to set the timestep for the particular intermolecular potential and integration algorithm used and the state of the fluid in question [130].

8.2.1 Stability: preventing energy cascades

The steep repulsive nature of the intermolecular potentials often used in MD simulations places an upper limit on the timestep that can be used. With reference to

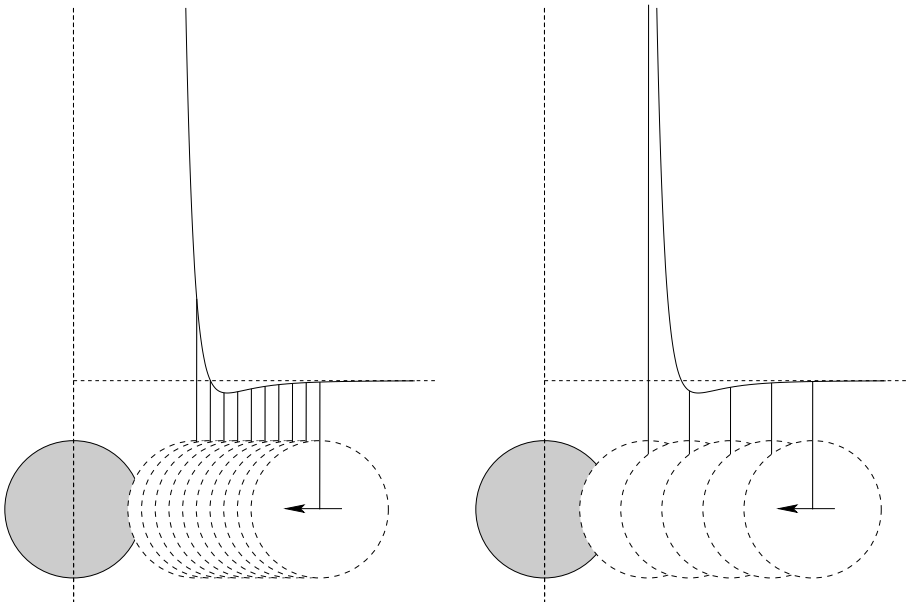


Figure 8.1: The approach of two molecules showing them tracking along the Lennard-Jones potential. When timesteps are short (left), the molecules repel smoothly. When timesteps are longer (shown here increased by a factor of three, right) the incoming molecule 'jumps' deep into the high energy repulsive region of the intermolecular potential, and will be repelled at a high velocity.

figure 8.1, consider two molecules approaching each other, and interacting with a Lennard-Jones potential. Considering motion relative to the grey molecule, the incoming molecule moves in finite 'jumps' in position, the size of which are determined by the velocity of the molecule and the timestep used for integration. When the timesteps are fine, then the intermolecular potential is sampled regularly, and when the repulsive (negative gradient) portion of the potential is reached, the incoming molecule is decelerated smoothly, then accelerated in the opposite direction: the molecules repel. When, however, the timesteps are large, the 'jump' in position when entering the repulsive region takes the incoming molecule deep into the repulsive portion of the potential. In the next timestep both molecules experience a very large repulsive force, accelerating each other to very high velocities. These high-velocity molecules then travel further into the repulsive region of the next molecules they collide with, resulting in twice as many molecules travelling at even higher speeds. This creates a cascade process and the kinetic energy of the molecules in the system rapidly diverges. The maximum velocity of molecules in the simulation, therefore the temperature, determines the maximum timestep.

8.3 Tracking molecules in arbitrary unstructured meshes

Tracking the motion of molecules in a simple geometry represented by a structured computational mesh does not present significant conceptual or computational difficulties: an algebraic expression can be used to determine which cell a particle occupies and when it encounters a boundary. In conventional MD a molecule can be moved along the trajectory for its current timestep without regard to an underlying mesh; it discards the information about which cell it previously occupied, then easily and quickly determines which cell it occupies in its new position for the purposes of force calculation.

When, however, the geometry is complex, comprising a mesh of unstructured, arbitrary polyhedral cells, as applied here, discarding and redetermining which cell a particle is in is no longer computationally efficient because it requires a time-consuming search of the mesh. It is more efficient to carefully track where and when particles cross mesh faces and change cell, and whether they hit boundaries as they move along their trajectory. Maintaining information about which cell a particle occupies is necessary for tracking the particle in subsequent timesteps and is also used when calculating intermolecular forces.

The particle base class in OpenFOAM provides robust motion tracking in arbitrary meshes (which may be moving or deforming), handles particles hitting any type of boundary (including periodic), and transfers particles between processors immediately as they hit interprocessor patches. The remainder of this chapter describes this generic particle[1] tracking algorithm, and is adapted from reference [3]. The molecule class used in gnemdFOAM is derived from the particle base class and inherits all of this tracking functionality automatically.

8.3.1 Basic particle tracking algorithm

Consider the situation in figure 8.2, where a particle is located at position **a** and is required to move to **b**, determined by solving the equation of motion for the particle. The motion can be performed as a series of individual tracking events, each ending when the particle either crosses a face of a cell, or arrives at the final destination. The particle must move to position **p**, where the line **ab** intersects

[1] "Particle" rather than "molecule" tracking will be referred to to emphasise the generality of the algorithm.

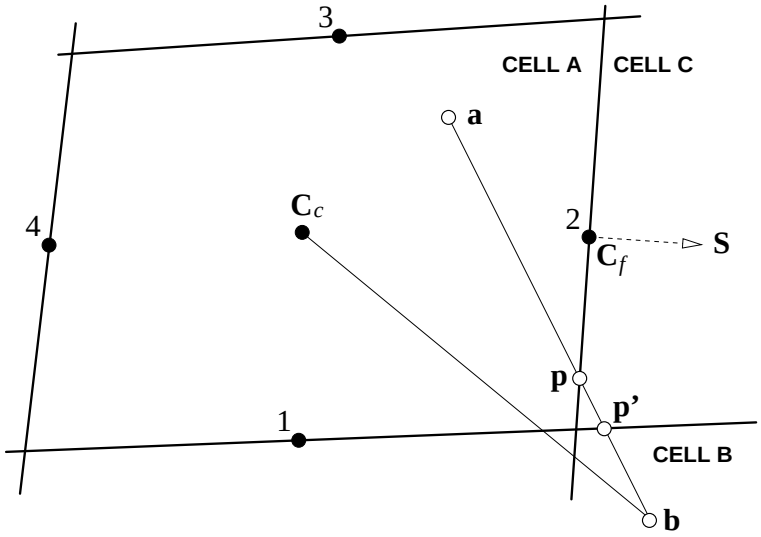


Figure 8.2: A particle moving from position **a** to **b**, crossing two faces at **p** and **p**′ and changing cell twice. The cell comprises four numbered faces, each of which stores face centre, $\mathbf{C}_f$, and face normal, **S**, vectors. The cell centre, $\mathbf{C}_c$, is shown.

face 2, then change to the neighbouring cell. It will carry on to **p**′, change cell again, before finally arriving at **b**. For the first part of the motion, **a** to **p**, the position **p** is found using

$$\mathbf{p} = \mathbf{a} + \lambda_a \left(\mathbf{b} - \mathbf{a}\right), \tag{8.12}$$

where λ_a is the fraction along the line **ab** where the intersection occurs with the plane defined by a face centre, $\mathbf{C}_f$ and face normal vector, **S**. Therefore, because **p** lies on this plane,

$$(\mathbf{p} - \mathbf{C}_f) \cdot \mathbf{S} = 0. \tag{8.13}$$

Substituting equation (8.12) into equation (8.13) gives,

$$\lambda_a = \frac{(\mathbf{C}_f - \mathbf{a}) \cdot \mathbf{S}}{(\mathbf{b} - \mathbf{a}) \cdot \mathbf{S}}. \tag{8.14}$$

Equation (8.14) is applied to calculate a value of λ_a for *each* face of the cell that the particle currently occupies, using each face's own $\mathbf{C}_f$ and **S** vectors. The face that the particle actually crosses is that which has the lowest value of λ_a in the interval $0 \leq \lambda_a \leq 1$. For the example shown in figure 8.2, faces 1 and 2 have λ_a values in this interval, with face 2 having the lower value, giving the correct face to be crossed.

The particle is moved to **p** and the particle's cell occupancy information is

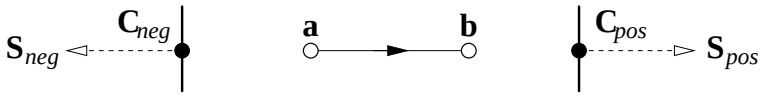


Figure 8.3: The physical interpretation of the value of λ_a. Face *neg* yields a negative value of λ_a, face *pos* yields a value of λ_a greater than 1.

changed to the neighbouring cell using the mesh connectivity: cell A, where the particle started, crosses face 2, cell A shares face 2 with cell C, therefore, the particle changes occupancy to cell C. Using this connectivity information means that computationally expensive mesh searching is avoided when maintaining cell occupancy information. The next tracking event is then executed in the same way: λ_a is calculated for each of the faces of the new cell, and the particle is tracked to the next face it has to cross, or to the final destination, if that is in the new cell.

If no face satisfies $0 \leq \lambda_a \leq 1$, then **b** must be inside the same cell as the particle started in, and it can be moved directly to the end point of the motion. For a particular face, $\lambda_a < 0$ corresponds to the particle motion from **a** to **b** being away from the face; $\lambda_a > 1$ corresponds to the motion from **a** to **b** being towards the face, but not reaching it. These two cases are shown in figure 8.3.

Deficiencies of the basic algorithm related to non-planar cell faces

Each face in the mesh stores a vector describing its face centre position and face normal vector; these define the plane of the face used when calculating if a particle crosses it. Where a face comprises more than three vertices, all of the vertices will not necessarily lie on a single plane. Therefore, the mesh stores face centroid and integrated area normal vectors which represent the *effective* plane of the face. Non-flat faces lead to a representation of the mesh that is no longer a set of space-filling cells. An example of this can be seen in figure 8.4, where face 2 has an exaggerated twist; plane 2 is the effective plane based on the centroid and integrated area normal vector. Plane 2 does not meet exactly with face 1, which is perfectly planar in this example.

There is a possibility of losing track of particles when they cross a face close to a vertex. In figure 8.4, a particle moving from **a** to **b** or **b′** will cross face 1, and, by mesh connectivity, change cell. The particle will, however, as it crosses face 1, be located physically on the wrong side of plane 2 to be consistent with

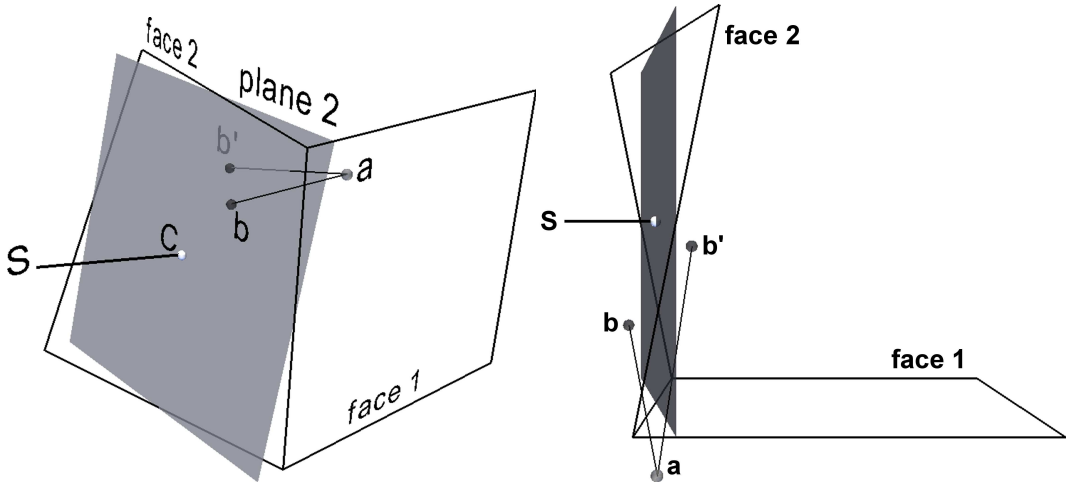


Figure 8.4: A mesh with a twisted face (face 2) which differs from its effective plane (plane 2). The figure on the right is a top-down view of the figure on the left.

occupying this new cell. The implications of this are clearer when considering the situation shown in figure 8.5, which is a 2D representation showing a situation *analogous* to that encountered during an event such as that shown in figure 8.4. Figures 8.4 and 8.5 are not directly related because it is only possible for the non-planar face issue to occur, and be represented, in 3D.

In figure 8.5 imagine that the particle has crossed face 1, leaving cell D, and, by mesh connectivity, its cell occupancy information is set to cell A. However, the position, $\mathbf{p}$, that the particle has tracked to lies outside of cell A. How the basic tracking algorithm responds depends on the final destination of the particle. The final destination can be either:

case b: inside the cell that is physically consistent with its current position (cell C). Values of λ_a calculated using $\mathbf{p}$ and each face of cell A (the cell that the particle's cell occupancy information states it should be in) are $\lambda_a < 0$ or $\lambda_a > 1$, which is equivalent to the particle not needing to change cell. The particle will move to $\mathbf{b}$, but it will not change its occupancy information from cell A to cell C.

case b′: inside the cell that the particle's cell occupancy information has been set to by mesh connectivity during the previous tracking event (cell A). The values of λ_a calculated using $\mathbf{p}$ and each face of cell A will determine that face 2 is intersected and the particle will change its occupancy information from cell A to cell C, but will be tracked into the physical region of cell A.

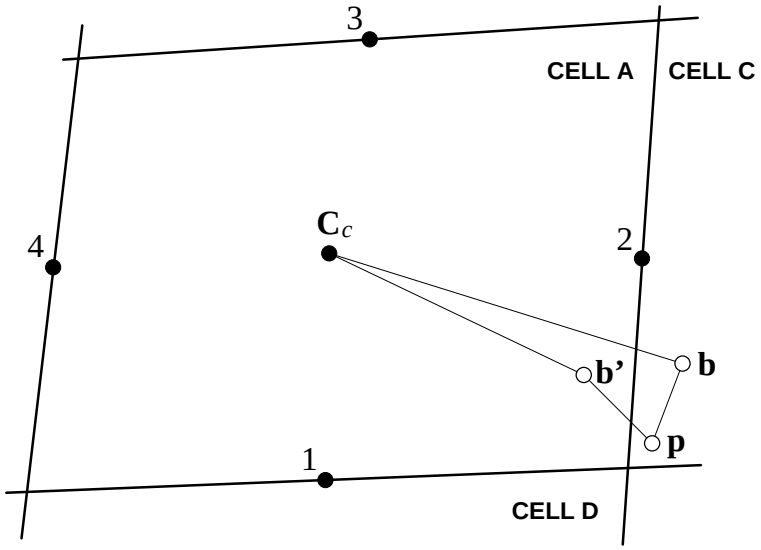


Figure 8.5: An analogous situation to a particle tracking across a face close to a vertex, in the vicinity of a non-planar face. The particle's post tracking event position, **p**, is located physically outside of the cell determined by connectivity information in the previous tracking event: cell A in this case.

In either case, the particle is not physically located in the cell that its cell occupancy information states it should be, and is effectively lost to the simulation. Correcting this problem would require a computationally expensive mesh search every time a particle changes cell in order to check that it moved into the cell it should have, and finding it if not.

8.3.2 Modified tracking algorithm

With reference to figure 8.2, if **b** is outside the starting cell, then a particle may be tracked from any position inside the cell and will cross the planes of the same faces that it would have if it had started at **a**. Therefore, rather than using **a** to find which face the particle will hit when moving from **a** to **b**, the cell centre, $\mathbf{C}_c$, can be used to calculate λ_c by replacing **a** with $\mathbf{C}_c$ in equation (8.14),

$$\lambda_c = \frac{(\mathbf{C}_f - \mathbf{C}_c) \cdot \mathbf{S}}{(\mathbf{b} - \mathbf{C}_c) \cdot \mathbf{S}}. \tag{8.15}$$

Using λ_c to find which face the particle will hit will result in $0 \leq \lambda_c \leq 1$ for face 1 and 2 in figure 8.2, as before. If $\lambda_c < 0$ or $\lambda_c > 1$ for all faces, then again as before, **b** must be inside the cell. It is necessary to calculate λ_a for all faces whose

Algorithm 4 Complete tracking algorithm.

```
while the particle has not yet reached its end position at b do
  find the set of faces, F_i for which 0 ≤ λ_c ≤ 1
  if size of F_i = 0 then
    move the particle to the end position
  else
    find face F ∈ F_i for which λ_a is smallest
    move the particle according to equation (8.16) using this value of λ_a
    set particle cell occupancy to neighbouring cell of face F
  end if
end while
```

planes are crossed (where $0 \leq \lambda_c \leq 1$, faces 1 and 2 in this case) and the lowest value of λ_a determines which face was actually hit. This value of λ_a is stored for use in the remainder of the calculation. The complete algorithm is summarised in algorithm 4.

The final components of the modified algorithm, and the reasons for the modification, become clear when returning to the two cases shown in figure 8.5:

case b: face 2 produces $0 \leq \lambda_c \leq 1$, but $\lambda_a < 0$, meaning that the **pb** trajectory points away from face 2. Here the particle is not moved, but the cell occupancy change determined by λ_c is performed, i.e. face 2 has been crossed. The particle changes occupancy from cell A to cell C and continues tracking to **b**. If the **pb** trajectory had pointed towards face 2 (but **b** was still in cell C), then $\lambda_a > 1$, and the particle is moved to **b** and the cell occupancy is changed to cell C as above. In both situations this is implemented by moving the particle to **p**, given by

$$\mathbf{p} = \mathbf{a} + \lambda_m (\mathbf{b} - \mathbf{a}), \quad \lambda_m = \min(1, \max(0, \lambda_a)); \tag{8.16}$$

case b′: all values of $\lambda_c < 0$ or $\lambda_c > 1$, so the particle is moved to $\mathbf{b}'$ and the cell occupancy stays as cell A.

It is possible to reduce the problems created by non-planar faces by decomposing each face into triangles and applying the basic algorithm to each of these subfaces [153]. This is, however, computationally more expensive than the modified algorithm above, and in practise, particles are still lost from the simulation due to numerical rounding errors, especially in moving meshes (see below). Rounding

errors can give rise to the same issue as encountered with non-planar faces, i.e. the particle's position is inconsistent with its cell occupancy information. The modified algorithm accommodates this, and as such rounding errors do not cause tracking failures or problems.

Concave cells

There is a possibility that the modified algorithm can enter an infinite loop if the cell it occupies is concave. Considering the example in figure 8.6. As the particle moves from **a** to **b** it does not need to leave (concave) cell A. It will, however, calculate an intersection with the plane defined by face 1 at **p**, and, by mesh connectivity, change occupancy to cell B. It is, however, moving away from face 2 in cell B, and will change occupancy to cell C, where it is moving away from face 3, and will change back to cell A and the process will start again, generating an infinite loop. Meshes using this algorithm must have concave cells decomposed into smaller convex cells.

Moving meshes

The algorithm as described is able to track particles in meshes that are moving by simply altering how λ_a is calculated, provided that the mesh does not move too far in a single step relative to the particle. Assuming that the mesh is moved before the particle tracking occurs, the mesh must not move so far as to place the particle more than one cell away from the cell it started in. It is assumed that particle tracking timesteps are short in comparison to the rate of mesh motion, and as such the motion of the mesh during the timestep can be assumed to be at a constant velocity. Figure 8.7 shows a single tracking event (not necessarily a full timestep) where a particle is attempting to track from **a** to **b** across a face in the mesh that is moving from a cell centre, face normal pair, $\mathbf{C}_s, \mathbf{S}_s$, at the start of the tracking event, to an end state, $\mathbf{C}_e, \mathbf{S}_e$. The particle will intersect the moving face at **p**, when the face is part-way through its motion at $\mathbf{C}_m, \mathbf{S}_m$. Given that the velocity of the particle and the linear and rotational velocities of the face are all considered constant throughout the tracking event, and that the mesh and particle are moving over the same time interval, then the same value of λ_a applies to determining the face centre and normal vectors at intersection as

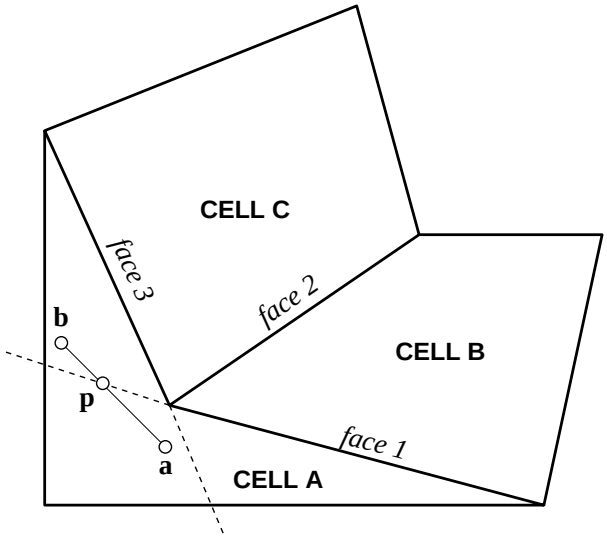


Figure 8.6: A particle moving in concave cell A from **a** to **b** will become stuck in an infinite loop.

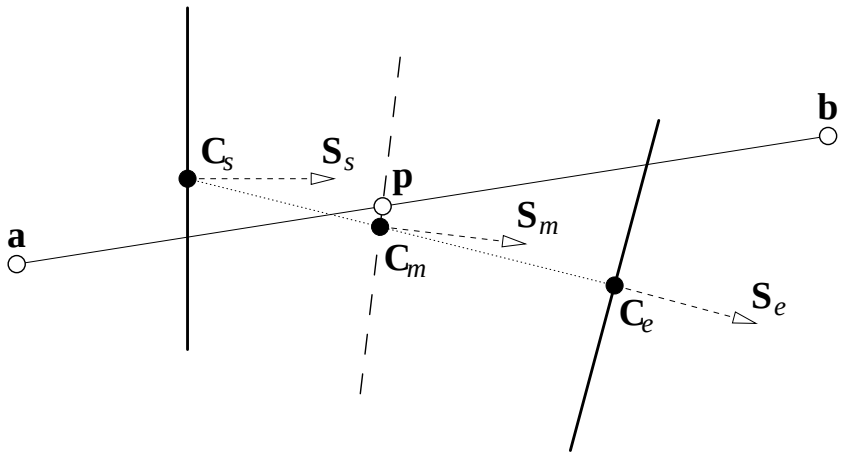


Figure 8.7: A particle tracking across a face that is translating and rotating.

for the particle motion in equation (8.12), i.e.

$$\mathbf{C}_m = \mathbf{C}_s + \lambda_a \left(\mathbf{C}_e - \mathbf{C}_s\right), \tag{8.17}$$

$$\mathbf{S}_m = \mathbf{S}_s + \lambda_a \left(\mathbf{S}_e - \mathbf{S}_s\right). \tag{8.18}$$

Substituting equations (8.17) and (8.18) for $\mathbf{C}_f$ and $\mathbf{S}$ into equation (8.14) results in a quadratic in terms of λ_a,

$$A_2\lambda_a^2 + A_1\lambda_a + A_0 = 0, \tag{8.19}$$

where

$$\begin{aligned} A_2 &= \big((\mathbf{b} - \mathbf{a}) - (\mathbf{C}_e - \mathbf{C}_s) \big) \cdot (\mathbf{S}_e - \mathbf{S}_s), \\ A_1 &= \big((\mathbf{b} - \mathbf{a}) - (\mathbf{C}_e - \mathbf{C}_s) \big) \cdot \mathbf{S}_s + (\mathbf{a} - \mathbf{C}_s) \cdot (\mathbf{S}_e - \mathbf{S}_s), \\ A_0 &= (\mathbf{a} - \mathbf{C}_s) \cdot \mathbf{S}_s. \end{aligned}$$

Equation (8.19) can either have two positive, real roots, in which case the root with the smallest magnitude is chosen, or imaginary roots, meaning the face was not intersected, so an (arbitrary) value of $\lambda_a > 1$ is returned. If the mesh undergoes linear motion only then $\mathbf{S}_e = \mathbf{S}_s$ and equation (8.19) reduces to

$$\lambda_a = \frac{-(\mathbf{a} - \mathbf{C}_s) \cdot \mathbf{S}_s}{\big((\mathbf{b} - \mathbf{a}) - (\mathbf{C}_e - \mathbf{C}_s) \big) \cdot \mathbf{S}_s}. \tag{8.20}$$

This process is equally valid for finding λ_c by substituting equations (8.17) and (8.18) into equation (8.15).

8.3.3 Boundary interactions and parallelisation

At every face crossing, a check is performed to determine if the face forms part of a boundary or is internal to the mesh. Specific actions can be taken depending on the type of boundary encountered, for example:

- periodic boundary: the particle is physically moved to 'wrap around' to the appropriate cell on the other side of the cyclic boundary, then continues the tracking step;
- interprocessor boundary: the particle is removed from the current processor

and recreated in the appropriate cell on the destination processor, where it completes the remainder of its motion;

- solid wall: the velocity of the particle is altered according to the wall model employed, e.g. specularly or diffusely reflected, see chapter 9. The particle now travels towards a different destination;
- outlet: the particle is deleted and removed from the simulation.

After a particle encounters a periodic boundary or a solid wall then, due to a position or velocity change, the particle tracks towards a different destination in the next tracking event than the one to which it was moving to at the beginning of the timestep. This is compatible with the algorithm as described.

Chapter 9

Boundary conditions and driven flows

Periodic and interprocessor boundaries are familiar concepts in MD. There are, however, more general types of boundary, as well as driving forces, required in order to provide the facility to perform flexible fluid mechanics simulations. Only simple solid walls (section 9.1) have been implemented and tested in the present code; the remaining sections in this chapter outline planned developments.

9.1 Simple solid walls

A simple planar surface is created by modifying the velocity of a molecule when it comes into contact with a patch of type 'wall.' The molecule can be specularly or diffusely reflected. This type of boundary can only be used in portions of the domain where the detailed molecular nature of the surface is not important, otherwise an explicit crystalline wall is required.

For a specular reflection, the component of the molecule's incident velocity normal to the face on the patch that it has collided with is reversed. If the incoming molecule's velocity is $\mathbf{v}$ and the face normal unit vector is $\mathbf{n}$ (which always points out of the domain, i.e. away from the incoming molecule) then the post-impact velocity, $\mathbf{v}'$ is

$$\mathbf{v}' = \mathbf{v} - 2\left(\mathbf{v} \cdot \mathbf{n}\right)\mathbf{n}. \tag{9.1}$$

For a diffuse reflection, the components of the post-impact velocity will be drawn

from a Gaussian distribution at a specified temperature T_D, in a similar way to section 6.6 where the velocities of newly created molecules are set. The unit vectors specifying the direction of the two components tangential to the wall, $\mathbf{t}_1$ and $\mathbf{t}_2$, are found using

$$\mathbf{t}_1 = \frac{\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\,\mathbf{n}}{|\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\,\mathbf{n}|}, \tag{9.2}$$

$$\mathbf{t}_2 = \mathbf{n} \times \mathbf{t}_1. \tag{9.3}$$

A normal distribution random number generator, Rn, which produces a series of independent values, Rn_j, with zero mean and variance $\sigma^2 = T_D/m_i$, where m_i is the mass of the incident molecule, is used to assign magnitudes to the components of $\mathbf{v}'$,

$$\mathbf{v}' = Rn_1\mathbf{t}_1 + Rn_2\mathbf{t}_2 - |Rn_3|\,\mathbf{n}. \tag{9.4}$$

The magnitude of the random number generated for the normal component is taken to ensure that the molecule moves back into the domain.

A 'statistical' wall is created by using a random number generator to give a fraction P_S of the molecules incident on it a specular reflection (where $0 \leq P_S \leq 1$) and $(1 - P_S)$ of the molecules a diffuse reflection. If $P_S = 0$ then all molecules are fully equilibrated with the wall, and the no-slip velocity boundary condition results.

9.2 Boundary force walls

A simple planar boundary, whether it is one that allows mass to pass through or not, creates an unphysical environment for proximate molecules because they do not experience intermolecular interactions with molecules that would be on the other side of the boundary. This creates an effective boundary condition of empty space on the other side of the boundary. In order to control the state of the fluid near the boundary (to give the correct density or pressure for example) by replacing these missing interactions, many authors [83, 114, 116, 119, 127, 154, 155] use boundary force models. These impose an artificial force on molecules that varies with distance away from the wall. They are often used in hybrid simulations

as part of the coupling mechanism between the continuum and molecular domains. One feature to note is that boundary force models can be used to create a region of depleted density near to the terminating edge of the domain to facilitate the insertion of particles into dense fluids.

9.3 Generalised open boundaries

A simple fluid outlet can be created by using a patch of type 'patch' that simply deletes a molecule when it comes into contact with it. A more general open boundary can be created that can control the mass, momentum and energy flux across it, and the state of the fluid near it. Such control cannot easily be exercised at each timestep, but over a number of timesteps average values may be imposed. The patch must:

- accommodate molecules impacting on it from inside the simulation domain;
- introduce molecules from outside of the domain.

When a molecule hits the patch, it has a probability, P, that it will:

- P_S: specularly reflect the molecule;
- P_D: diffusely reflect the molecule with a velocity corresponding to a temperature of T_D;
- P_R: allow the molecule to pass through, it is then removed from the simulation;

where $P_S + P_D + P_R = 1$.

The patch can attempt to insert molecules of a particular species, k, at a specified mass flowrate per unit area, R_{Ik} with a velocity corresponding to a specified T_I (which may or may not be the same as T_D) and bias velocity $\mathbf{v}_I$. The molecules can be introduced into the system at a target potential energy, U_I using published algorithms [128, 129].

The insertion of molecules will happen on a face-by-face basis (this allows the insertion energy, temperature and bias velocity to vary across the patch: OpenFOAM's mesh description allows the properties of the patch to be easily spatially varied. This could be used to impose a velocity profile on the flow at an

inlet or an arbitrary temperature profile along a wall. Mass must be introduced in discrete values corresponding to a whole number of molecules. One strategy for achieving this is to create a mass counter, C, for each species and the mass to be introduced through the face in question can be accumulated onto it. For a face of area A, which is to introduce mass at a rate of $R_{Ik}\, kg/m^2 s$ of a particular species, then, for a timestep of duration Δt,

$$C_k^t = C_k^{t-1} + R_{Ik} A \Delta t. \tag{9.5}$$

For a constant rate of introduction C will grow linearly. At each timestep, the value of C_k/m_k will be examined for each counter, where m_k is the mass of one molecule of the species in question. If

- $C_k/m_k \leq 0$ then no attempt will be made to introduce molecules in this timestep;
- $0 < C_k/m_k < 1$ then there is a probability of C_k/m_k that an attempt will be made to introduce a molecule at this timestep — a uniform random number generator will be used to determine this. If the molecule is successfully inserted then C is decremented by m_k (hence negative values of C_k/m_k can occur, see figure 9.1). If the insertion of the molecule fails (for example, a suitably low energy position near the face cannot be found) then the mass is retained on the counter and another attempt will be made at a later timestep. The use of random numbers leads to a stochastic introduction of flux, which seems more physical than insertion at a constant interval;
- $C_k/m_k \geq 1$ then $\lfloor C_k/m_k \rfloor$ molecules will have insertion attempts made and C is decremented by m_k for each that succeeds. There is a further probability of
$$C_k/m_k - \lfloor C_k/m_k \rfloor$$
that another attempt is made, as for the case above when $0 < C_k/m_k < 1$.

If too many molecule insertions fail (most likely due to high fluid density near to the patch) then C will continue to accumulate mass that will never be inserted. In this case a boundary force wall can be incorporated to create a low density region to improve insertion rates.

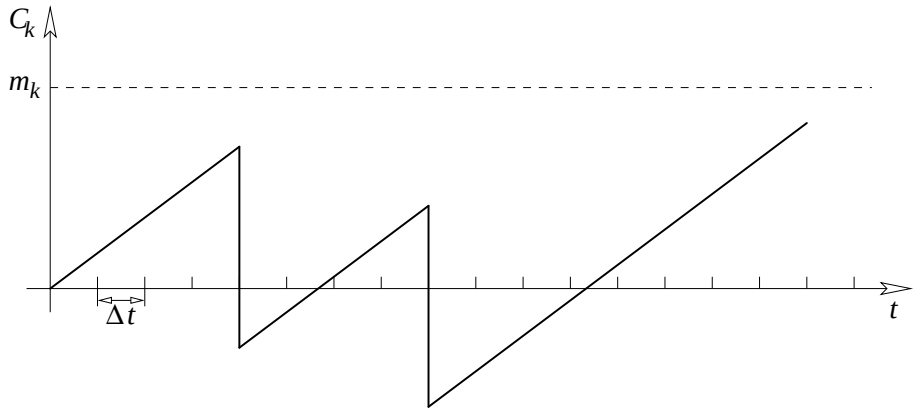


Figure 9.1: The mass addition counter for a face and species, C_k, climbs linearly and particle insertions occur with a probability of C_k/m_k at each timestep. When a particle is inserted C_k is reduced by m_k.

9.3.1 Variable permeability periodic boundaries

This framework may be used to create periodic patches with a probability of 'wrapping' the molecule around the boundary or reflecting it. This can be used to create a pressure gradient to drive a flow [156]. This can be achieved by suppressing molecule insertion ($R_I = 0$) and using probabilities such that the molecule hitting the patch will

- P_S: specularly reflect the molecule;
- P_D: diffusely reflect the particle with a velocity corresponding to a temperature of T_D;
- P_W: be wrapped around the periodic boundary.

Where $P_S + P_D + P_W = 1$ as before.

9.4 Thermostats, constraints and driven flows

It is conventional in MD to use thermostats and constraints in order to drive a system to a desired state and sample a specific statistical mechanical ensemble [130, 157]. These operate by either modifying the Hamiltonian of the system, or by coupling the molecules to an external 'heat-bath' and statistically modifying their velocities. The system then no longer simulates purely Newtonian dynamics.

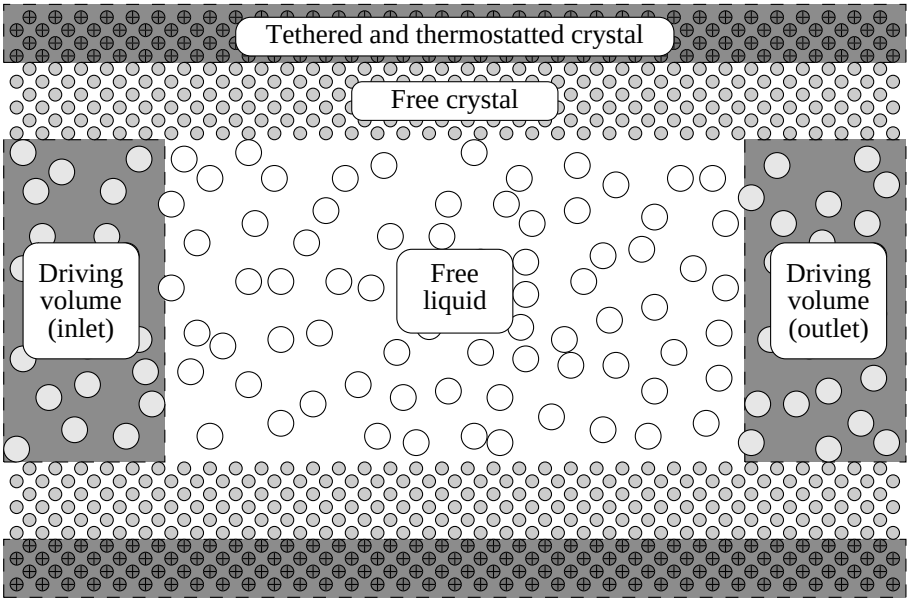


Figure 9.2: A simple system employing driving volumes (shaded regions) and wall molecules tethered (molecules with crosses) into a crystal to constrain the fluid and solid boundaries of a system. Only the free regions, where natural Newtonian dynamics occur, will be used to generate results. The free wall crystal will remain a solid by virtue of its intermolecular potential only.

Control of the state of the fluid in fluid mechanics simulations is carried out at boundaries only, to allow the dynamics in the test section to exhibit natural (momentum and energy conserving) dynamics. To accomplish this, volumes of the mesh will be defined as thermostatted or constrained sections, called driving volumes (see figure 9.2). These can be combined with tethering molecules to ensure that solid walls stay stationary or move with a prescribed average velocity, see figure 9.2. Molecules inside the driving volumes experience the required modifications to their properties and revert to undergoing natural dynamics when they leave it. The thermostats and constraints applied inside driving volumes can vary spatially by using the underlying mesh to set their parameters, allowing arbitrary profiles of velocity or temperature to be imposed, for example.

There are several aspects that are common to applying thermostats and constraints to conventional MD simulations that cannot be applied in arbitrary geometries. In particular:

- conventional methods for controlling pressure in particular involve expanding and contracting the MD simulation domain itself to modify the interatomic spacing. These cannot be employed in complex domains, although selective removal of molecules to reduce the overall density may be feasible;

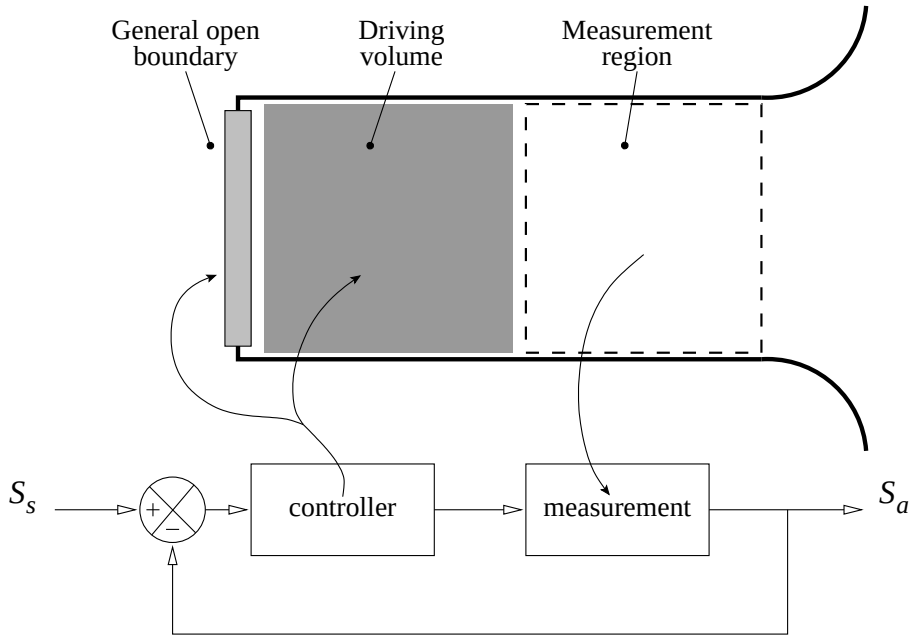


Figure 9.3: A fluid inlet or outlet created using feedback control. The set points, S_s, for the properties to be controlled are compared to the actual values, S_a, in a measurement region. A controller adjusts the parameters of a boundary and driving volume.

- the measurement and control of molecules must depend only on local properties, not the global state of the system. Any mechanism that depends on knowing the peculiar velocity of the molecules (i.e. the molecule's velocities with the bulk streaming velocity subtracted) will present a problem. It is difficult to accurately determine the streaming velocity with fine spatial and temporal resolution, see section 10.4, so methods that do not require this must be used [158–160].

The driving volume infrastructure will also be useful in coupling continuum and molecular systems together in a hybrid simulation, because overlap regions, where the dynamics of molecules are constrained, are often used [114, 116].

9.5 Setting and feedback control of parameters

The generalised boundary model (section 9.3) has $7 + K$ adjustable parameters to control the patch, where K is the number of species involved: P_S, P_D, P_R, R_{Ik}, T_D, T_I, U_I and $\mathbf{v}_I$. Average mass flowrate can be controlled by varying the insertion and removal rates, P_R and R_{Ik}. The energy and momentum flux, however, are linked through the velocity assigned to molecules (controlled by T_D

and T_I), as well as the insertion and removal rates, ratio of specular to diffuse reflections and U_I, the potential energy new molecules are introduced with. The state of the fluid will also depend on all of the parameters in a complex manner. It is therefore unlikely that a desired condition can be achieved by simply setting the parameters at the start of a simulation.

A solution to this would be to control the parameters using a feedback loop. For example, if the pressure at the inlet to a channel is to be controlled, then a region downstream of the inlet can have its pressure measured and a feedback loop used to control the patch parameters to a achieve desired value. Creating a driving volume for fluid inside the domain near the boundary and controlling its parameters in a similar way can be combined with control of the boundary to achieve the desired state. Figure 9.3 shows a sketch of this scheme. Deducing the response of the system to patch parameter variations, determining the number of properties that can be simultaneously controlled, and designing the controller to be applied would be a significant challenge.

Chapter 10

Spatially resolved flow properties

The capabilities described in chapters 6 to 9 allow molecules to be simulated moving through the volume defined by a mesh. In order to make the simulation useful, measurements of the system need to be made. As with continuum CFD, the cells in the mesh will hold the information about the state of the fluid in the volume they define. Measurements are required of:

- the state of the fluid in the cell, represented by an average value assigned to the cell centre;
- transport through the cell, represented by the net flux crossing the cell faces.

Typically the properties of the molecules contained in a cell are averaged at any given timestep, and many timesteps are averaged to produce useful results. These measurements are required

- to give an indication of the state of the fluid and flow for its own analysis;
- for a hybrid simulation, to provide data to couple the MD and continuum components together.

10.1 Spatial and temporal averaging in cells

Choosing the resolution of the mesh and the averaging period requires a balance of competing aims. Long averaging periods lead to a better signal-to-noise ratio (all properties are subject to fluctuations) but can mask the details of the evolution

of a time-varying system. Likewise, small cells show fine spatial resolution of properties, but require longer averaging periods to increase the number of samples in the measurement to reduce noise. There is also a dependence between the resolution of cells in the mesh required for property measurement and that needed to optimise the speed of intermolecular force calculation. If the mesh needs to be finer for property measurement than for force calculation, then there is no problem because excess cells do not have a substantial negative impact on the force calculation performance (see chapter 12). If the mesh needs to be finer for force calculation than property measurement then properties are measured on the finer mesh and the data from several cells can be aggregated and 'smoothed' in a post-processing stage. This requires that no simplifications or assumptions are made regarding the functional form of the data, number of samples or averaging periods; 'raw' data will be collected. The measurement techniques described are designed to remain valid for ≤ 1 molecule per cell and single timestep resolution (no time averaging). Fixed duration timesteps are assumed; variable timesteps would require appropriate weighting when constructing averages.

It is often the case that the full 3D field of a system is not of interest, but the average variation of properties in one direction. For example, consider a system with periodic boundaries in two directions (say x and y) and solid walls bounding the third (z), with a variation of a property in the z direction. The simulation is intended to examine the dynamics of the 1D variation, and as such measurements will be made by averaging across x-y planes (providing there are no appreciable 3D flow patterns). For the purposes of intermolecular force calculation, there will need to be many cells in each x-y plane, whereas it would be convenient for measurement to have only one cell per plane. Again, data will be collected in the finer cells, and calculating plane averages will occur as a post-processing step, ensuring that the contributions from unequally sized cells are appropriately weighted.

If the dynamics of a system are too fast to permit long enough time averaging periods to provide a suitable signal-to-noise ratio, then ensemble averaging may be used. The system can be started in several different but equivalent configurations (the same initial molecular positions with the same initial temperature, but a different set of random numbers for the velocities, for example) and the results combined. Care must be taken to identify if the dynamics of the systems diverge, and are no longer following the same trend.

There will be exceptions to the case above where fine cells required for measurement can be tolerated for force calculation. When calculating radial distribution functions [130, 131] or the electron density profile of a system to compare it to x-ray experiments [66] then very fine resolution histograms are required (1pm bin widths for the latter) which would require an unmanageable number of cells. These cases will be handled by creating a set of cells comprising the volume to be measured. A histogram will be created with the required resolution corresponding to variation in a user defined direction. Alternatively, the variation as a function of distance normal to a set of faces on the boundary of the cell set could be used.

10.2 Measureable properties

The spatial and temporal variation of a large number of physical properties of a fluid and flow can be measured:

- *average velocity*;
- *kinetic temperature*;
- configurational temperature;
- *mass density* and *mass fraction*;
- *number density* and *mole fraction*;
- *velocity distribution*;
- pressure;
- stress;
- heat flux;
- entropy;
- mass, momentum and energy flux across a surface;
- radial distribution function [124, 130, 131];
- correlation functions [124, 130, 131];
- transport coefficients (e.g. viscosity, thermal conductivity or diffusivity).

All variables can be measured on a per-species basis for multi-species simulations and on a total basis. For example the density of a single species compared to the total density gives composition of the fluid, or in highly non-equilibrium situations different species may have different effective temperatures.

Measurement of the properties in italics has been implemented in the present code. The measurement of the others has presently not been implemented be-

cause either there is ambiguity or debate about the method or meaning of the measurement, particularly in an arbitrarily shaped volume, or a need has not arisen for the data in the cases studied. The measurement of stress in particular is contentious; literature exists describing stress tensor measurement [161–163][1], but its definition is not unique [169]. The infrastructure for defining and measuring a field in gnemdFOAM is general, so a new measurement may be rapidly implemented when the simulation requires it, or to experiment with competing definitions.

10.2.1 Gas collision properties

Molecular dynamics can be used to test the limits of kinetic theory and investigate rarefied and non-equilibrium gas dynamics (for more discussion see appendix F). The following can be measured in addition to the properties outlined above to investigate this:

- probability of a molecule being simultaneously in collision with a given number of other molecules;
- distribution of free paths between collision;
- distribution of time that molecules spend in collision.

The spatial variation of these averaged collision properties can give an indication of the non-equilibrium processes occurring in rarefied gases, particularly in the vicinity of solid walls.

10.3 Velocity

For a system containing K different species of molecules, a running sum is kept of the momentum of each species and the total mass of each species in each cell. Data is accumulated for each timestep, t, of the averaging period that is τ timesteps long. The mass of a molecule of particular species is m_k. It is the same for all molecules of that species and constant for all timesteps in the averaging period.

The average velocity of the fluid in a cell is given by the total momentum of the molecules divided by the total mass of molecules in the cell [170, 171], i.e.

[1] The same authors are responsible for many other relevant publications in the non-equilibrium MD field, i.e. [13, 110, 164–168]

$$\overline{\mathbf{v}} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} m_{itk}\mathbf{v}_{itk}}{\sum_{k=1}^{K}\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} m_{itk}}, \tag{10.1}$$

where m_{itk} and $\mathbf{v}_{itk}$ are respectively the mass and velocity of a molecule i, of species k, at timestep t. Given that the mass of a molecule of a particular species is constant for all timesteps, then m_{itk} becomes m_k and

$$\sum_{i=1}^{N_{tk}} m_k = m_k \sum_{i=1}^{N_{tk}} 1 = m_k N_{tk}. \tag{10.2}$$

Let N_k be the total number of molecules of species k in the cell over all τ timesteps,

$$N_k = \sum_{t=1}^{\tau} N_{tk}. \tag{10.3}$$

Substituting these definitions into equation (10.1) gives

$$\overline{\mathbf{v}} = \frac{\sum_{k=1}^{K} m_k \sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}}{\sum_{k=1}^{K} m_k N_k}. \tag{10.4}$$

A mean velocity for the k^{th} species is defined as

$$\overline{\mathbf{v}}_k = \frac{1}{N_k}\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}. \tag{10.5}$$

10.4 Kinetic temperature

Defining the average kinetic temperature of the fluid in a cell as (by extending equation 21 in [168], ignoring any terms accounting for the loss of degrees of freedom caused by constraints):

$$\overline{T} = \frac{1}{3k_b N}\sum_{k=1}^{K} m_k \sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} (\mathbf{v}_{itk} - \overline{\mathbf{v}}) \cdot (\mathbf{v}_{itk} - \overline{\mathbf{v}}) \tag{10.6}$$

where k_b is the Boltzmann constant,

$$N = \sum_{k=1}^{K} N_k,$$

is the total number of molecules of all species in the cell, and $\overline{\mathbf{v}}$ is the mean, or 'streaming' velocity. The temperature is therefore given by the mean kinetic energy of the molecules, where their peculiar (or thermal) velocities are used to measure this. The use of $\overline{\mathbf{v}}$ is problematic in two ways. The first problem is a practical computing concern: the sum as shown cannot be formed on a timestep-by-timestep basis and stored as a running sum because $\overline{\mathbf{v}}$ is not known until the end of the averaging period. It would be possible to record all velocities in each cell and construct the peculiar velocities from them at the end of the averaging period once the streaming velocity is known. This would require, however, a prohibitive amount of memory, especially for large systems using long averaging periods. Fortunately equation (10.6) can be rearranged to avoid this; expanding the dot product:

$$\begin{aligned}\overline{T} &= \frac{1}{3k_bN}\sum_{k=1}^{K} m_k \sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \left(\mathbf{v}_{itk}\cdot\mathbf{v}_{itk} + \overline{\mathbf{v}}\cdot\overline{\mathbf{v}} - 2\mathbf{v}_{itk}\cdot\overline{\mathbf{v}}\right), \\ &= \frac{1}{3k_bN}\sum_{k=1}^{K} m_k \left(\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}\cdot\mathbf{v}_{itk} + \sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \overline{\mathbf{v}}\cdot\overline{\mathbf{v}} - 2\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}\cdot\overline{\mathbf{v}}\right),\end{aligned}$$

and noting that $\overline{\mathbf{v}}$ does not depend on i or t, so can be brought out of these sums:

$$\overline{T} = \frac{1}{3k_bN}\sum_{k=1}^{K} m_k \left(\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}\cdot\mathbf{v}_{itk} + N_k\overline{\mathbf{v}}\cdot\overline{\mathbf{v}} - 2\overline{\mathbf{v}}\cdot\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}\right). \qquad (10.7)$$

It is enough to add $\mathbf{v}_{itk}$, $\mathbf{v}_{itk}\cdot\mathbf{v}_{itk}$ and N_{tk} to running sums for each cell to allow $\overline{\mathbf{v}}$ and $\overline{T}$ can be calculated at the end of the averaging period, after τ timesteps.

The second problem surrounding the use of $\overline{\mathbf{v}}$ will occur when there are insufficient samples to give an accurate streaming velocity. Large uncertainty will remain in the measured average velocity, which will lead to inaccuracy in measurements, like kinetic temperature, that depend on it. This will be a particular problem for time varying systems using fine cells.

The temperature for a single species in a mixture can be defined as

$$\overline{T}_k = \frac{m_k}{3k_b N_k}\left(\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}} \mathbf{v}_{itk}\cdot\mathbf{v}_{itk} + N_k\overline{\mathbf{v}}\cdot\overline{\mathbf{v}} - 2\overline{\mathbf{v}}\cdot\sum_{t=1}^{\tau}\sum_{i=1}^{N_{tk}}\mathbf{v}_{itk}\right), \tag{10.8}$$

although it is not clear if the streaming velocity in this case should be the all-species velocity $\overline{\mathbf{v}}$, or the streaming velocity for the individual component $\overline{\mathbf{v}}_k$. These will only be significantly different under highly non-equilibrium conditions, or when there are only a small number of samples to compute an accurate average velocity.

Given that the subtraction of the streaming velocity is used to essentially remove the effect of a moving reference frame on the measurement of temperature, it would seem that the all-species streaming velocity, $\overline{\mathbf{v}}$ should be used. In this case, the total temperature can be calculated from the average of the single species temperatures weighted by the relative abundance of the species,

$$\overline{T} = \sum_{k=1}^{K}\frac{N_k}{N}\overline{T}_k. \tag{10.9}$$

Note that using the definition of temperature given by equation (10.6), where the streaming velocity is removed, involves an implicit assumption of local thermodynamic equilibrium in the cell [172].

An alternative and more intuitive way of considering the total temperature for a multi-species mixture comes by writing equation (10.6) with the mass of each molecule included in the sum and the (1/2) retained in the kinetic energy expression:

$$\overline{T} = \frac{2}{3k_b N}\sum_{i=1}^{N}\frac{1}{2}m_i\left(\mathbf{v}_i - \overline{\mathbf{v}}\right)\cdot\left(\mathbf{v}_i - \overline{\mathbf{v}}\right). \tag{10.10}$$

A single summation incorporating each molecule at a timestep and all timesteps in the averaging period, N molecules in total, has been used for clarity. Similarly, the average velocity is given by,

$$\overline{\mathbf{v}} = \frac{\sum_{i=1}^{N} m_i\mathbf{v}_i}{\sum_{i=1}^{N} m_i}.$$

Defining variables for the total mass,

$$M = \sum_{i=1}^{N} m_i,$$

and momentum

$$\mathbf{P} = \sum_{i=1}^{N} m_i \mathbf{v}_i,$$

so,

$$\overline{\mathbf{v}} = \frac{\mathbf{P}}{M},$$

and expanding the dot product in equation (10.10) to give:

$$\overline{T} = \frac{2}{3k_b N}\left[\left(\sum_{i=1}^{N} \frac{1}{2} m_i \left(\mathbf{v}_i \cdot \mathbf{v}_i\right)\right) - \frac{1}{2}\frac{(\mathbf{P}\cdot\mathbf{P})}{M}\right].$$

The first term in brackets is the internal kinetic energy of the fluid, the second term is the kinetic energy associated with its bulk motion.

10.4.1 Other definitions of temperature

When a fluid is out of equilibrium, there is no longer a unique, unambiguous definition of temperature. Alternative definitions, such as the 'configurational temperature' can be used [173–176]. It has the benefit of not requiring the streaming velocity to be calculated first [158, 173].

10.5 Density and mass/mole fraction

The single species number density, ρ_{Nk}, in a cell is given by the average number of molecules of species k per timestep divided by the cell volume V,

$$\rho_{Nk} = \frac{N_k}{\tau V}, \tag{10.11}$$

and the total number density, ρ_{Nk}, is given by

$$\rho_N = \frac{N}{\tau V} = \sum_{k=1}^{K} \rho_{Nk}. \tag{10.12}$$

The mole fraction, ϕ_{Nk}, of species k is given by

$$\phi_{Nk} = \frac{N_k}{N} = \frac{\rho_{Nk}}{\rho_N}. \tag{10.13}$$

Similarly, the single species mass density, ρ_{Mk} is given by the average mass of species k per timestep divided by the cell volume V,

$$\rho_{Mk} = \frac{m_k N_k}{\tau V}, \tag{10.14}$$

and the total mass density, ρ_M, is given by,

$$\rho_M = \frac{1}{\tau V} \sum_{k=1}^{K} m_k N_k = \sum_{k=1}^{K} \rho_{Mk}. \tag{10.15}$$

The mass fraction, ϕ_{Mk}, of species k is given by

$$\phi_{Mk} = \frac{m_k N_k}{\sum_{k=1}^{K} m_k N_k} = \frac{\rho_{Mk}}{\rho_M}. \tag{10.16}$$

10.6 Velocity distribution

The Maxwellian 'velocity' distribution is in fact a molecular speed distribution, constructed by each component of velocity having a normal distribution [145]. There is no preferred direction of motion and a single temperature defines the width of the distribution, hence the magnitude of the velocity components. When the fluid is flowing, and not in equilibrium, it is not particularly useful to form the distribution of molecular speeds. The distribution of velocity in a particular component direction, and the differences between distributions in different component directions, is more informative. These directions can vary from cell to cell, and be supplied externally, or self generated. For example, at a surface, the velocity distribution component normal to the local surface, the distribution aligned with the local streaming velocity, and the remaining transverse direction can be measured. The directions will vary from cell to cell but will always represent the normal, flow and transverse components of the flow. The structure of the mesh and fields in OpenFOAM makes this easy; the distribution for a particular cell can be easily linked to the geometrical properties of the cell or its constituent

faces, or any of the field data stored for that cell.

10.6.1 Distribution class

The measured distribution of speed is stored in an object of the 'distribution' class (essentially a histogram) which is constructed using a user specified value for the width of its histogram bins, b_w. There is no need to specify the number of bins because the underlying data structure dynamically resizes itself to the correct number. When a value is added to the distribution then n, the key for the bin that the value lies in is calculated, and the count variable, $D(n)$, for that bin is incremented by one. If the key does not already exist, it is created and its count set to 1. Keys are always integers to avoid floating point comparisons. A key of value n represents a bin of width b_w, centred at $(n+1/2)b_w$. The appropriate key for a scalar quantity v is given by

$$n = integer(v/b_w) - neg(v/b_w), \tag{10.17}$$

i.e. taking the integer part[2] of v/b_w and correcting for negative values by using the boolean function $neg()$ which returns 1 if its argument is negative and 0 otherwise.

When the distribution is to be output the keys are converted to their equivalent bin centre values and the accumulated counts must be normalised to $D'(n)$, such that:

$$\sum_{n=n_{min}}^{n_{max}} D'(n)b_w = 1. \tag{10.18}$$

The sum of all of the bin counts gives the total number of values recorded, N_v,

$$\sum_{n=n_{min}}^{n_{max}} D(n) = N_v, \tag{10.19}$$

[2] $integer()$ is not the floor or ceiling function because it always rounds towards zero, for example, $integer(3.2) = 3$, $integer(-5.8) = -5$.

which (because N_v does not depend on n) rearranges to,

$$\frac{1}{N_v}\sum_{n=n_{min}}^{n_{max}} D(n) = 1,$$

$$\Rightarrow \sum_{n=n_{min}}^{n_{max}} \frac{1}{N_v b_w} D(n) b_w = 1, \tag{10.20}$$

so, by comparing equations (10.18) and (10.20), the normalised values are

$$D'(n) = \frac{1}{N_v b_w} D(n). \tag{10.21}$$

The distribution class is also able to calculate the mean and median of the data and 'shift' the distribution along the bin value axis by a user specified value, or by its own mean to centre the distribution. The shift applied to the distribution will generally not be a whole bin width, so the operation involves remapping the shifted data onto standard bin centres.

This class can be used as a template to measure other properties that require a 2D representation, for example, correlation functions or the structure factor of a fluid. Examples can be found in

- section 12.1.2. The distribution of the magnitude of intermolecular potential calculation errors caused by imperfect mesh searching using the PP and PPGR methods (figures 12.3 and 12.4) are recorded by the distribution class;
- appendix C. The probability distributions for maximum velocity in a timestep (figure C.2) were recorded, and had their mean values calculated by, the distribution class;
- appendix F. Results shown in all graphs (figures F.1, F.2 and F.3) were produced using a precursor to the distribution class in an older code, gnemd (see chapter 12).

Part III

Performance and testing

Chapter 11

Molecular configuration generation performance

A cubic test system of 50 units side length (in MD reduced units) was chosen to evaluate the performance of molConfig. The commercial CFD meshing software GAMBIT® was used to generate 21 different tetrahedral meshes with these dimensions. They were generated by specifying the number of cell edges to be placed on the edges of the bounding geometry, and then allowing GAMBIT® to automatically generate a tetrahedral mesh. Edge numbers of 10–30 were used and the number of cells, N_C, generated in each case are shown in table 11.1. An example mesh (edge number 15) is shown in figure 11.1. A single zone was defined for the whole mesh and filled with an SC lattice with five different densities, $\rho = \{0.8, 0.4, 0.2, 0.08, 0.02\}$, creating $N_M = \{97336, 46656, 27000, 10648, 2744\}$ molecules in each mesh. The time taken for molConfig to fill each mesh was recorded and is shown in figure 11.2. Linear trends fit the computational time data very well, although the negative intercepts of the fitted lines indicate that linear scaling does not hold when there are very few cells. All timing tests were performed on a PC with a 2.8GHz, AMD Athlon™FX-62 processor.

Each line in figure 11.2 represents a constant number of molecules with a varying number of cells in the mesh. The dependence of the computational time on the number of molecules is examined by holding the number of cells constant, as shown in figure 11.3 for three meshes. The three vertical lines on figure 11.2 correspond to the meshes chosen. This data does not fit a linear trend; the computational cost grows more slowly than a linear relationship with increasing number of molecules in a fixed mesh. A function of the form:

Table 11.1: Number of cells in test tetrahedral meshes generated by GAMBIT®.

edge no.	10	11	12	13	14	15	16	17	18	19
N_{cells}	7520	9748	11101	19344	22289	22338	28113	36646	43420	50877
20	21	22	23	24	25	26	27	28	29	30
56171	67520	73801	88334	97488	105385	129761	145016	145196	185016	196847

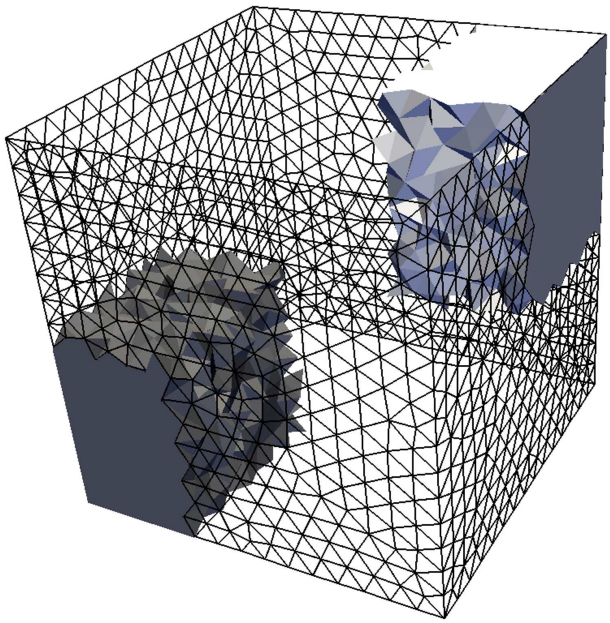


Figure 11.1: Tetrahedral test mesh with edge number 15. The outline of cells on four of the six faces of the bounding cube are shown, and two of the twelve portions of cells are shown from the mesh when decomposed for parallel processing.

$$p + q\,(N_M/r)^s\,,$$

where p, q, r and s are the fitting parameters, was found to fit the data well, with r and s similar for each line.

In a practical system, the computational cost will increase as approximately the 'problem size' squared, because a larger problem will involve a larger mesh with more cells, where the cost is proportional to N_C, as well as a larger number of molecules, where the cost is approximately proportional $N_M^{0.88}$.

11.1 Aspect ratio

The speed of molecule generation is highly dependent on the shape of the mesh; for the cubic lattices used here, cubic domains are the fastest to fill because they

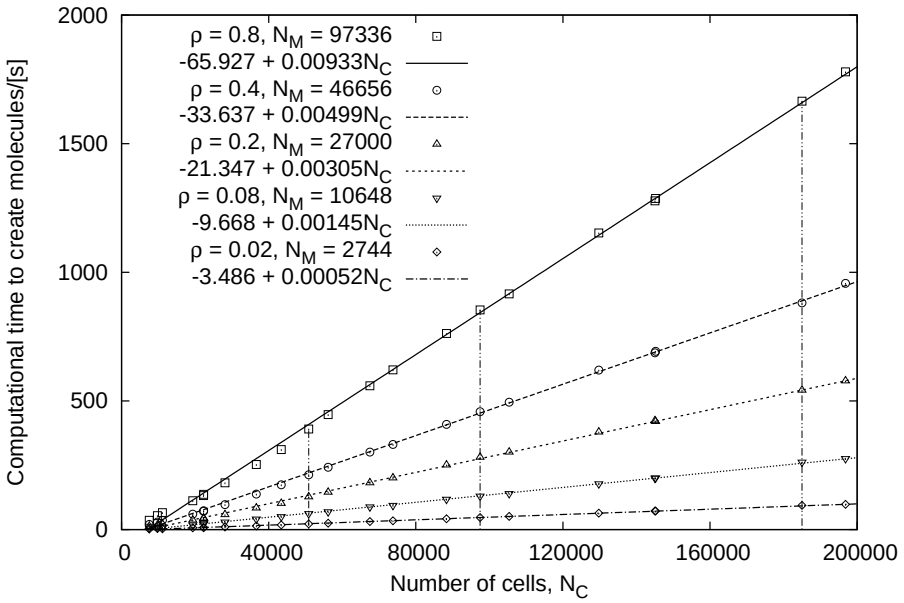


Figure 11.2: The computational time required to generate a given number of molecules with a varying number of cells in the mesh. The vertical lines are the meshes selected for figure 11.3.

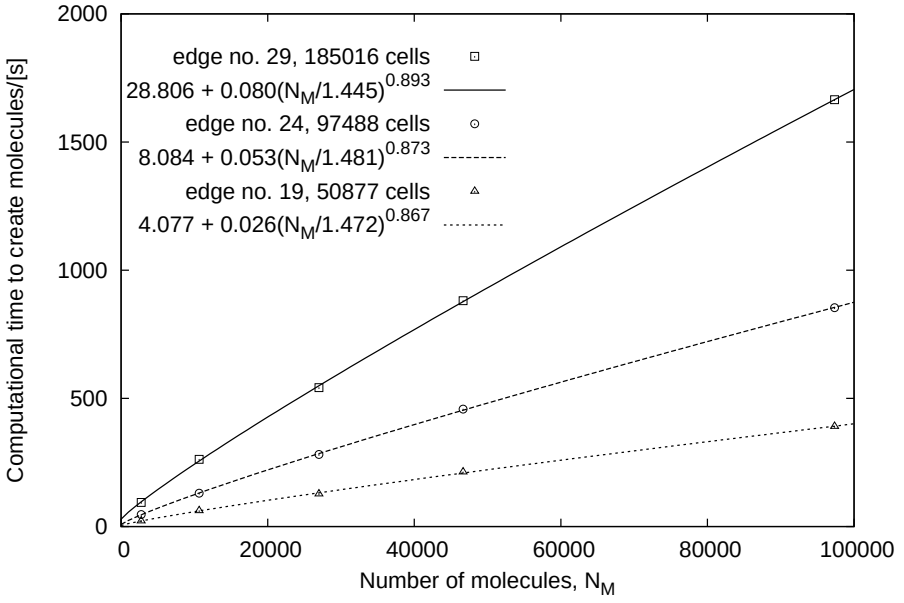


Figure 11.3: The computational time required to fill a mesh of a given number of cells with a varying number of molecules.

Table 11.2: The performance of molConfig for a cuboid system with varying aspect ratio.

B	40	38	36	34	32	30	28	26	24	22	20
a	1	1.166	1.371	1.628	1.953	2.370	2.915	3.641	4.630	6.011	8
N_M	64000	63888	63936	64220	64288	63900	64304	63580	63504	63534	64000
time/[s]	17	38	74	120	176	300	450	687	1186	1959	3433

match the shape of the expanding volume of unit cells. Any deviation from cubic takes longer to fill for the same number of cells and molecules. To examine this, a cuboid regular mesh of $20\times20\times20$ cells was created with various aspect ratios, see figure 11.4. The number of cells and the volume of the system was held constant to give approximately[1] the same number of molecules, N_M. For a cuboid with a base area of B^2, and a side length of L, the volume is

$$V = B^2 L. \tag{11.1}$$

Defining the aspect ratio, a, of the system to be

$$a = \frac{L}{B}, \tag{11.2}$$

substituting this into equation (11.1), and rearranging for a gives

$$a = \left(\frac{B_1}{B}\right)^3 \tag{11.3}$$

where B_1 is the side dimension when $a = 1$, used to set the total system volume. Here $B_1 = 40$, so $V = 64000$. Table 11.2 lists the aspect ratios used, the number of molecules generated and the computational time required to fill each of the meshes with an SC lattice of density = 1. The computational time taken for each aspect ratio, relative to the time required for $a = 1$, is shown in figure 11.5. A quadratic function fits the data very well. High aspect ratio mesh shapes incur a significant computational time penalty with the expanding cube volume filling-algorithm because unnecessary unit cells will be tested in a large volume outside the mesh.

[1]The number of molecules varies because the geometry for a particular aspect ratio does not exactly match the geometry of the lattice that it is being filled with. However, the maximum deviation from, in this case, 64000 molecules is only 496, or 0.775%.

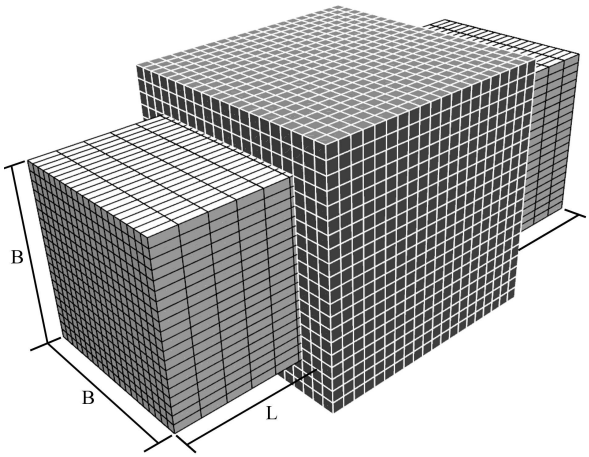


Figure 11.4: Two examples of the mesh used to examine aspect ratio sensitivity: $a = 1$ (dark solid with white grid) and $a = (40/28)^3 = 2.915$ (light solid with black grid). Note that the number of cells is the same in both cases (20 in each direction); they are stretched to match the overall dimensions.

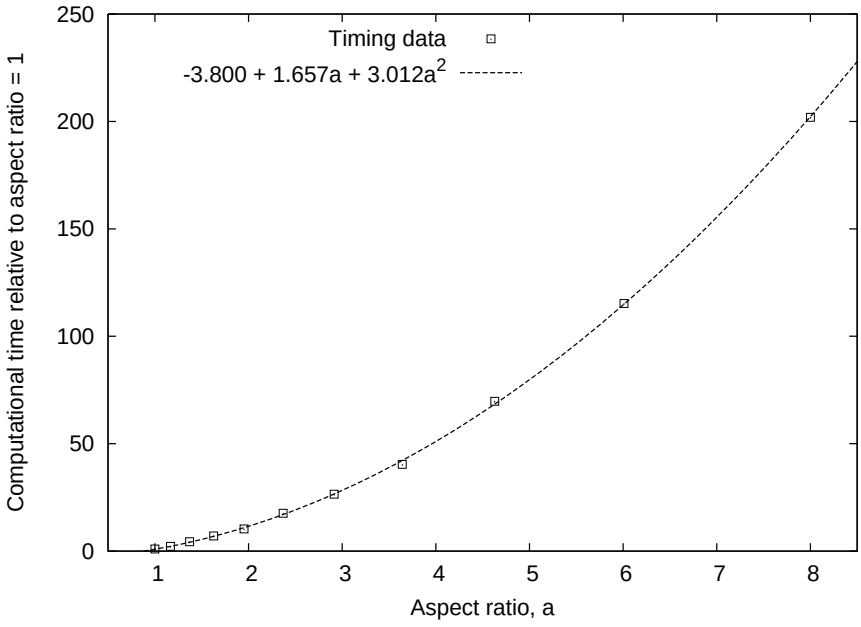


Figure 11.5: The computational cost of filling varying aspect ratio cuboids, relative to the cost of a cubic system of the same volume. A quadratic dependence is observed.

11.2 Parallel performance.

The parallel performance of molConfig has not been subjected to timing tests because the processors do not need to communicate during the filling process. Any deviation from ideal speedup for parallel molecule generation will be mostly determined by the size and shape of the portions of the mesh that result from spatial decomposition. For example, a cubic system divided into 8 smaller, equally sized cubes and filled on 8 processors is likely to fill faster than the same cubic system divided into 12 strips and filled on 12 processors.

11.3 Choosing mesh dimensions and anchor positions

The overall dimensions and refinement of the mesh will be usually dictated by concerns other than the speed of molecule generation: the impact on the speed of intermolecular force calculation and the resolution of flow properties. The scaling of the molConfig algorithm is linear with the number of cells in the mesh and better than linear with the number of molecules, so the molecule generation process does not impose unreasonable constraints on the choice of mesh. Very high aspect ratio shapes (long channels, for example) present a performance penalty, and should be subdivided into several smaller zones to speed up molecule generation. Placing the anchor for all of these zones at the same place will ensure that they are filled with a single crystal; the anchor will be optimised for each zone automatically.

Choosing the dimensions of the domain requires care if a solid crystal wall extends across a periodic boundary; the domain must be sized to permit an integer number of lattice unit cells in the wall zone to prevent gaps or overlaps in the lattice. The anchor of the lattice should be chosen so that a plane of a lattice does not lie exactly along the surface of the mesh. If this occurs, then numerical rounding errors will cause patchy placement of molecules in this plane — some molecules will be inside the domain and some outside. A small shift of the anchor will move the plane off the surface and prevent this. A similar issue occurs when generating molecules in parallel: if a molecule lies exactly on a face shared by two processors, each processor may place a molecule in essentially the same place. This is more difficult to prevent, because it depends on how the

mesh is decomposed, but does not cause problems because one of the duplicated molecules will be removed by the high energy overlap check.

Chapter 12

Intermolecular force calculation performance

12.1 Accuracy: average pair potential energy

The total potential energy of all pair interactions in a test system was calculated by an in-house code: gnemd. It was written in C++ and was based on and validated against the code supplied in reference [130]. The positions of all of the molecules from the gnemd test case were imported into gnemdFOAM, transferring with a precision of 18 significant figures. The same system (same bounding box geometry and same intermolecular potential) was simulated by gnemdFOAM using a regular cubic mesh and the 21 tetrahedral meshes as used in chapter 11. The cubic mesh in gnemdFOAM comprised $28^3 = 21952$ cubic cells. Periodic boundaries in each direction were applied.

A cubic system domain of exactly 50 units side length was chosen to avoid any round-off errors when generating the mesh geometry, and a fluid number density of $(50/46)^{-3} = 0.778688$ chosen to give a perfect simple cubic lattice of $46^3 = 97336$ molecules. The total potential energy of all pair interactions for the molecules in their initial configuration, divided by the number of molecules, will be used to compare the results. All MD values in these tests will be expressed in reduced units scaled using the argon Lennard-Jones potential length, energy and mass values, see section 5.1.3. A cut-off radius of $r_{cut} = 2.5$ was used in each case.

This average potential per molecule, $\langle PE \rangle$, was calculated using gnemd, $\langle PE \rangle_g$, and by gnemdFOAM for 22 meshes (21 tetrahedral and cubic) using

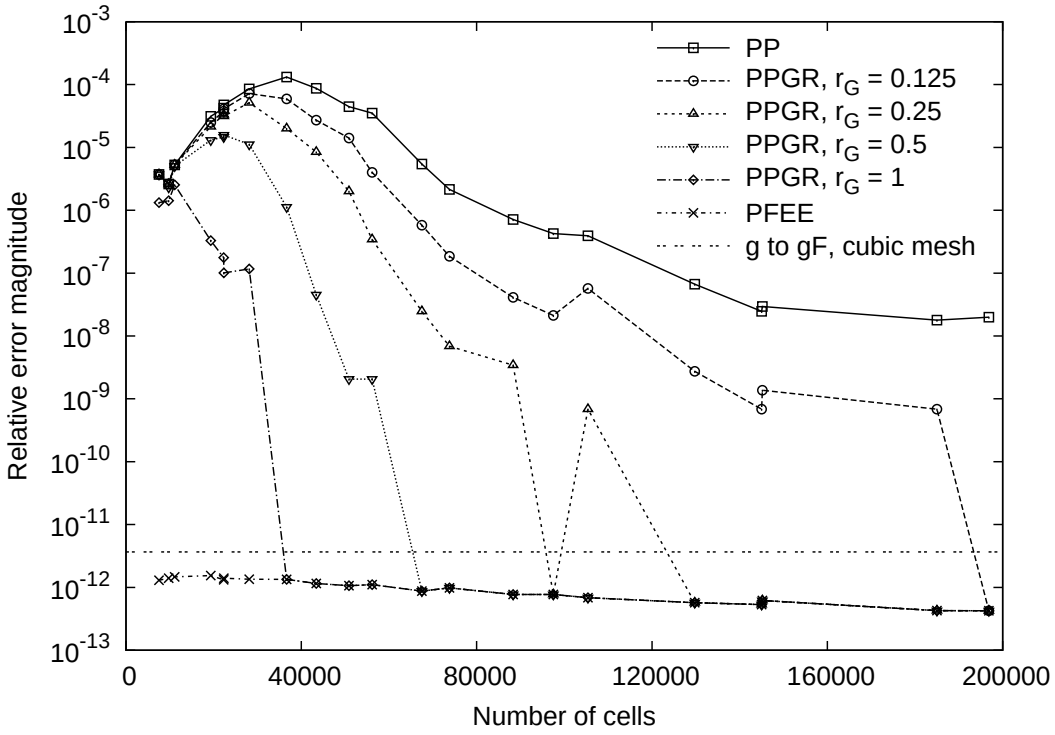


Figure 12.1: Relative error between $\langle PE \rangle$ for each tetrahedral mesh and $\langle PE \rangle_{gF}$ with different cell interaction build methods.

the PP, PPGR (with $r_G = 0.125, 0.25, 0.5, 1.0$) and PFEE cell interaction identification methods. The cubic mesh in gnemdFOAM was not sensitive to any of the mesh searching errors of PP identified in section 7.4 and produced exactly the same result, $\langle PE \rangle_{gF}$, with each of the cell interaction identification methods:

$$\begin{aligned} \langle PE \rangle_g &= -4.22656275761469, \\ \langle PE \rangle_{gF} &= -4.22656275759921, \\ \left| \frac{\langle PE \rangle_g - \langle PE \rangle_{gF}}{\langle PE \rangle_{gF}} \right| &= 3.64 \times 10^{-12}. \end{aligned} \tag{12.1}$$

The difference between $\langle PE \rangle_g$ and $\langle PE \rangle_{gF}$, equation (12.1), most probably arises from the accumulation of rounding errors in the calculation of all pair potentials[1] and any round-off error in the transfer of molecule positions between the two codes. The magnitude of this difference is considered to be small enough to indicate no algorithm or implementation errors. The relative error between $\langle PE \rangle$

[1]There are approximately 50 molecules in range of each other molecule, giving approximately $97336 \times 50/2 = 2433400$ pair interactions.

for the tetrahedral meshes using the various cell interaction methods and $\langle PE \rangle_{gF}$ is calculated in the same manner as equation (12.1) and shown in figure 12.1. The errors in the PP or PPGR results are due to cells containing molecules that should interact not being identified by point-point searching of the mesh. The errors in PP and PPGR simulations become smaller in meshes with more cells because a fixed $r_{cut} + r_G$ length becomes larger relative to the cell size, giving a greater chance of establishing two points of a cell being in range.

Note that PFEE always produces a result that appears to be on the numerical 'noise floor' of the simulation — where the cumulative effect of numerical rounding errors becomes important. This is supported by the PPGR results dropping onto this line when r_G is large enough, indicating that all errors have been removed. This line has a smaller magnitude than the value from equation (12.1), which is plotted as 'g to gF, cubic mesh'. This demonstrates that PFEE identifies the correct cells to provide every molecular interaction in the system, irrespective of mesh topology.

12.1.1 Parallel accuracy

The accuracy of PFEE when the simulation is parallelised is demonstrated in figure 12.2. The tetrahedral meshes were decomposed into 12 portions (as shown in figure 11.1) by a simple $x : y : z = 2 : 3 : 2$ split and into 32 portions using METIS [141], then simulated on a distributed memory parallel cluster. The 'parallel to serial error' data represents the relative difference between $\langle PE \rangle$ calculated for each mesh and $\langle PE \rangle_{gF}$. The 'parallel self error' data represents the difference between $\langle PE \rangle$ calculated for each mesh and $\langle PE \rangle$ calculated for the cubic mesh parallelised in the same way as the tetrahedral meshes. It is interesting to note that the parallel self errors are substantially lower than the parallel to serial errors; the cubic mesh simulated in parallel gives a result substantially closer to the parallel tetrahedral meshes than the serial result. The parallel to serial errors are larger than those for PFEE in figure 12.1, although still acceptably small. This demonstrates that the referred cells, which provide the intermolecular force connection between processors, are correctly created and used.

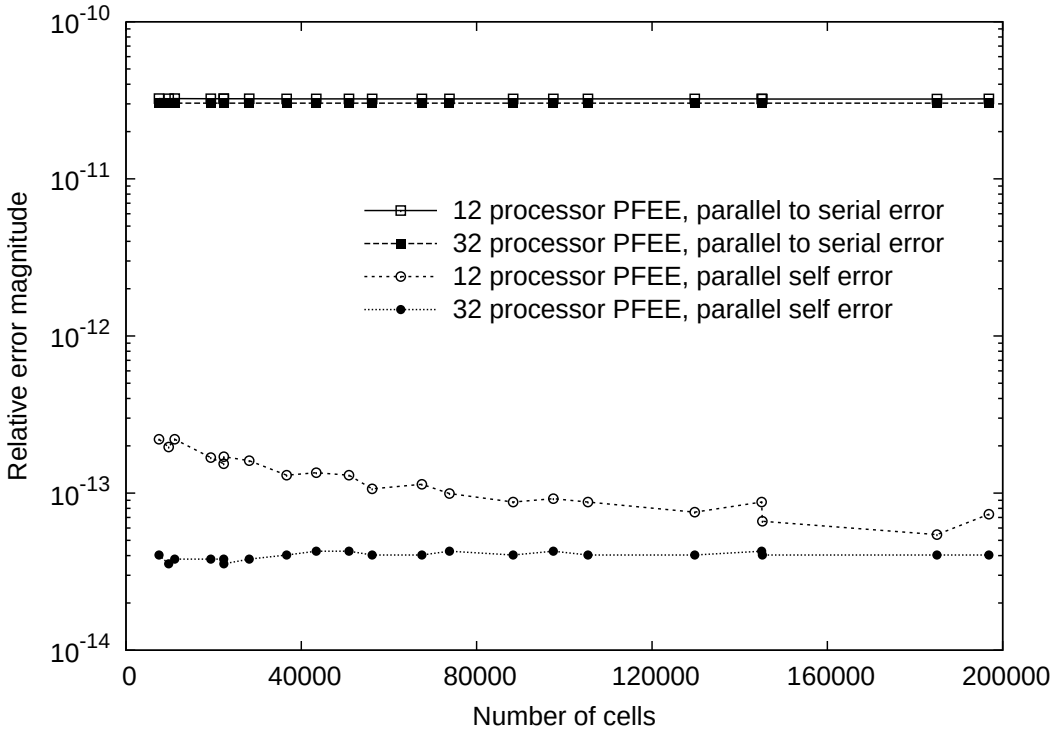


Figure 12.2: Relative error in $\langle PE \rangle$ for each tetrahedral mesh simulated in parallel using 12 and 32 processors. Cell interactions built with PFEE.

12.1.2 Distribution of errors

The values of $\langle PE \rangle$ for PP and PPGR do not give an indication of how the intermolecular force errors due to missed interactions are distributed. If a large number of molecules experience small errors, then this could be tolerable. If, however, a small number of molecules receive substantial errors, this will may cause unacceptable perturbations to the dynamics. The latter case is observed in practise.

The system simulated above is initialised in the cubic mesh with molecule velocities drawn from a Maxwellian distribution at a temperature of 2.5. The system is allowed to evolve in gnemdFOAM using the leapfrog integration scheme for 1000 timesteps of $\Delta t = 0.005$ to create a reference system. This was done to let the initial simple cubic lattice relax into a more realistic configuration, because the net force on each molecule in the perfect crystal is zero, which is not useful for calculating relative errors. The molecule positions from the reference system are used in the tetrahedral meshes and the acceleration vector for each molecule calculated. The absolute and relative error magnitude between the acceleration for each molecule in the tetrahedral mesh and the acceleration for

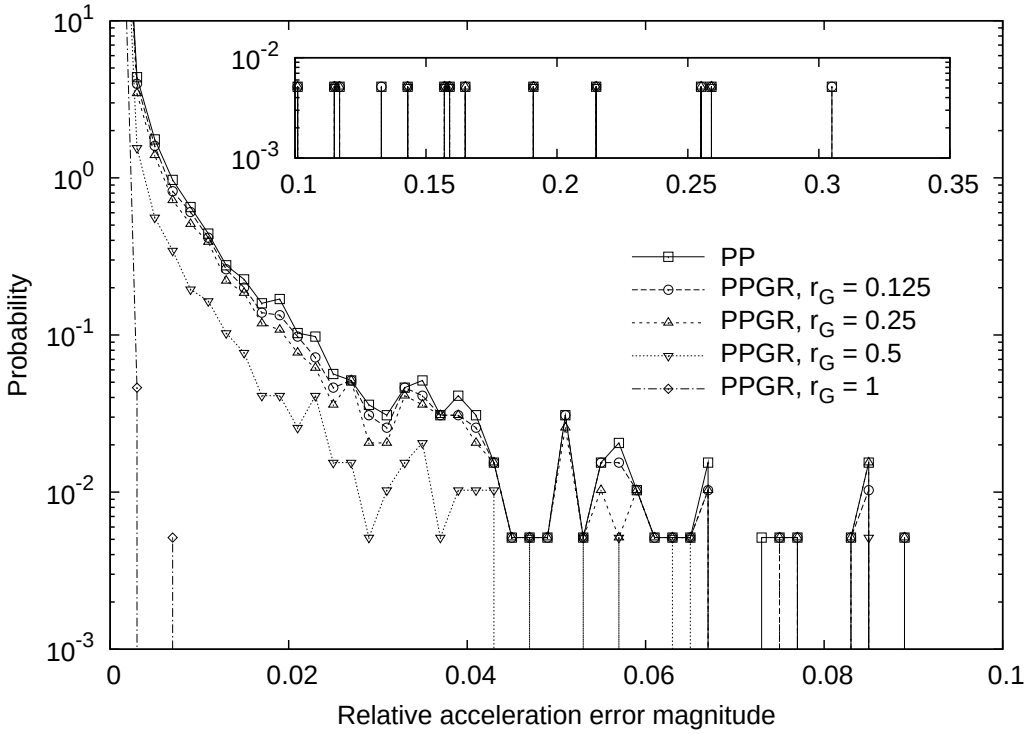


Figure 12.3: Relative error magnitude distribution for tetrahedral mesh, edge number 14.

the corresponding molecule in the reference system is calculated:

$$\text{absolute}_i = |\mathbf{a}_{\mathbf{tet}i} - \mathbf{a}_{\mathbf{ref}i}|\,, \tag{12.2}$$

$$\text{relative}_i = \frac{|\mathbf{a}_{\mathbf{tet}i} - \mathbf{a}_{\mathbf{ref}i}|}{|\mathbf{a}_{\mathbf{ref}i}|}. \tag{12.3}$$

These errors were added to histograms to assess their distribution. Examples of the histograms are shown in figures 12.3 and 12.4 for the edge number 14 mesh, using PP and PPGR ($r_G = 0.125, 0.25, 0.5, 1.0$). Both graphs show that the vast majority of molecules receive very small errors. A small number of molecules receive a substantial absolute error — up to 2.5, where acceleration magnitudes of up to 100–150 are typical in this simulation. Large relative errors of 30% (see the inset portion of figure 12.3) are also seen, although these may be caused by the reference acceleration magnitude being very small. The addition of a guard radius does improve matters, but not substantially until, in this example, $r_G > 0.5$ is used.

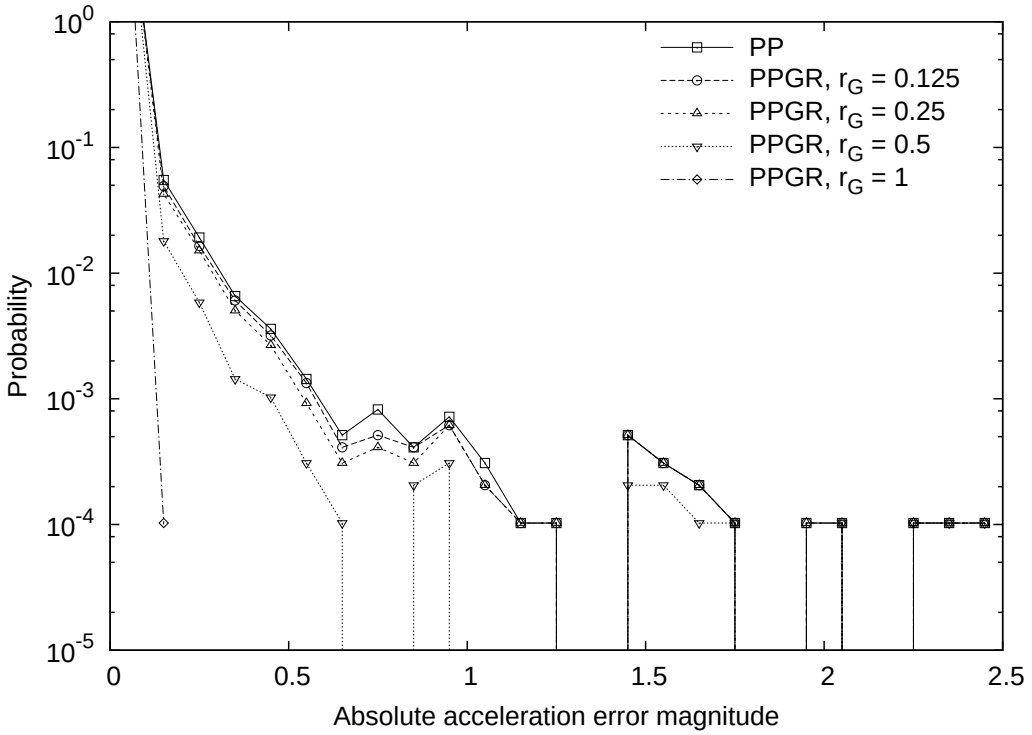


Figure 12.4: Absolute error magnitude distribution for tetrahedral mesh, edge number 14.

12.2 Computational speed

The computational speed of establishing cell interactions and calculating intermolecular forces in gnemdFOAM was assessed for each of the tetrahedral meshes above, filled with 97336 LJ molecules. The time taken to build the interaction lists (create DILs, create referred cells and find real cells in range of referred cells) was recorded, and an intermolecular force calculation for all molecules performed 400 times and timed. The molecules were not moved during the simulation. All timing tests in this section were performed on a PC with a 2.8GHz, AMD Athlon^{TM}FX-62 processor.

The time taken to build the interaction lists is shown in figure 12.5 (note the logarithmic time axis). For the PP and PPGR data, an increase with increasing r_G is seen because more referred cells will be created. It is clear that PFEE takes substantially longer than PP or PPGR to build the interaction lists. This is understandable given the number of comparisons and calculations required by PFEE. Building the interaction lists is $O(N^2)$, for a mesh of N cells, and as such benefits greatly from parallelisation because each portion of the mesh only

needs to search itself. The longest PFEE builds, taking several 1000s of seconds, would typically not be practical and the mesh would be decomposed and the simulation parallelised. The interaction lists are established only once, at the start of the simulation, so for a very long simulation this may not represent a substantial portion of the total time. If the same mesh with the same r_{cut} is to be used repeatedly, then the cell interaction information can be calculated once then written to, and read from disk.

The average time taken per timestep to calculate all intermolecular forces is shown on figure 12.6. All of the results follow a similar trend: in meshes with few cells, each cell contains a large number of molecules, and many pairs identified to interact will be further apart than r_{cut}. As the cells become smaller, the number of unnecessary interactions reduces until a point where the additional overhead of administrating more cells becomes more costly than the benefit derived, producing a 'dip.' All of the results essentially level off for meshes with a large number of cells (greater than 80000). This shows that the presence of empty cells (there are 97336 molecules in the system) does not present a significant overhead, a favourable characteristic. The only exception is for PPGR, $r_G = 1.0$, where the large number of additional interacting cells seems to cause a noticeable overhead increase in meshes containing many cells.

The PP and PPGR results produce a family of curves, with increasing r_G the computational time increases and the region of the 'dip' narrows. An increased guard radius will produce DILs with unnecessary cells on them and unnecessary real-referred cell interactions, increasing the number of intermolecular evaluations between pairs of molecules that are further apart than r_{cut}.

The PFEE result has a similar form to PP and PPGR but does not fit into the family. In meshes with fewer cells, PFEE is more expensive than PP and most of the PPGR results because it is correctly identifying cells to interact that PP and PPGR miss. Comparing figure 12.6 to figure 12.1 it can be seen that the 'dip' (up to approximately 40000 cells) in computational time coincides with the region of maximum errors for PP and PPGR — missed cell interactions emphasise the 'dip' by reducing the number of calculations required by each molecule. The fact that PFEE does not take significantly longer to calculate forces than PP for finer meshes demonstrates that it does not identify any more cells to interact than are absolutely necessary.

There is an opportunity to trade-off the time taken to build the interaction

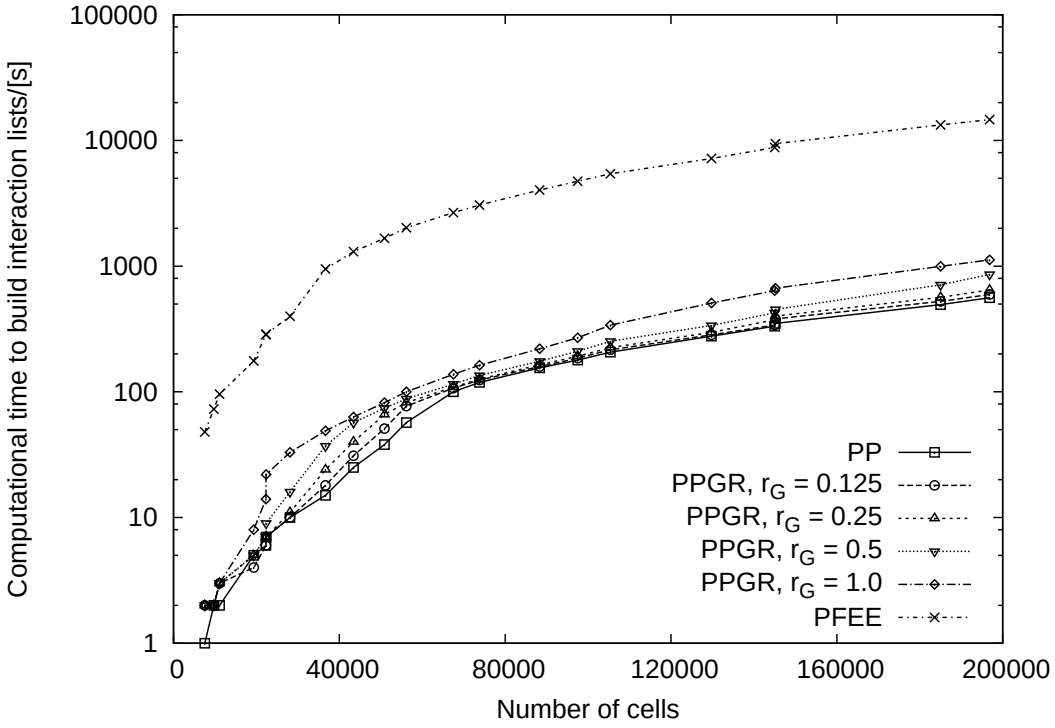


Figure 12.5: Time taken to build interaction lists at the start of the simulation.

lists using PFEE with a quick list build and slower timesteps using PPGR and a large enough r_G to remove all errors. The required r_G length is difficult to determine in advance, however, and most realistic simulations involve very many timesteps, which would soon make the PPGR simulation slower overall.

Further simulations to characterise the performance of the code, exploring the impact of molecule number density, system size, mesh cell size and morphology (for example, tetrahedral vs. hexahedral, skew, aspect ratio) number of processors and decomposition method are required.

12.2.1 Speed comparison with conventional MD code

To assess the speed of calculation relative to a conventional MD neighbour-list method, gnemdFOAM was compared to programs pr_04_1 and pr_04_2 from reference [130]. The test case was a cubic system of density $\rho_N = 0.8$, with periodic boundaries in each direction. A regular cubic mesh of varying resolution was created for the gnemdFOAM cases. Two sizes of system were considered, $N_M = 12^3 = 1728$ molecules and $N_M = 26^3 = 17576$ molecules. Each system size was simulated at two temperatures, $T = 1.0$ and $T = 2.5$, and two cut-off

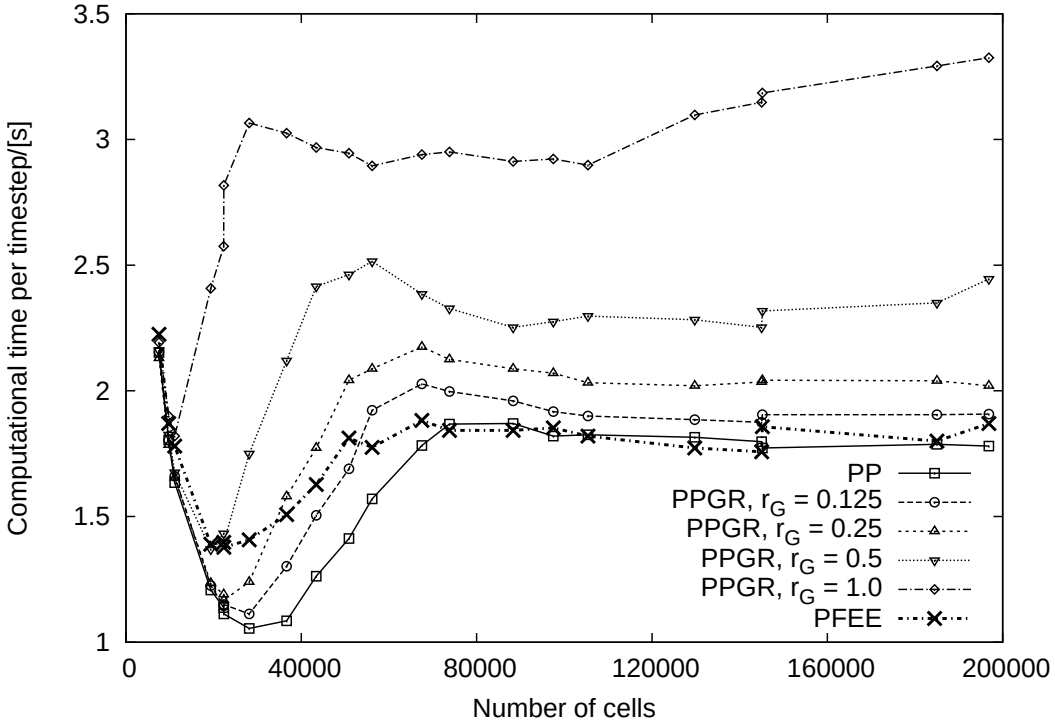


Figure 12.6: Time taken to calculate intermolecular pair forces between all molecules.

radii were used, giving eight permutations of simulation case. The purely repulsive WCA potential with a cut-off radius of $r_{cut} = 2^{1/6} \approx 1.122$ was used to compare to pr_04_1 and the unshifted, truncated Lennard-Jones potential with $r_{cut} = 2.5$ was used to compare to pr_04_2. For the cases with 1728 molecules, 10000 timesteps were simulated; for the 17576 molecule case 2000 timesteps were simulated.

The results are shown in tables 12.1 and 12.2. The number of cells in the gnemdFOAM meshes are shown along with the time taken for each case to run, t_{gF}, as well as the reference time, t_{ref}, taken by pr_04_1 or pr_04_2 to run. In the gnemdFOAM cases, only the time spent calculating timesteps (force calculation and integration of equations of motion) is shown. The time taken to build the interaction lists is excluded because it is a one-off cost at the start of the simulation and its relative impact depends on the length of the simulation. For each case the ratio between the fastest gnemdFOAM mesh (minimum t_{gF}) and the reference time is calculated,

$$R_{gF} = \frac{min(t_{gF})}{t_{ref}},$$

Table 12.1: Simulations comparing pr_04_1 ($r_{cut} = 2^{1/6}$) with gnemdFOAM.

no.	N_M	T	$t_{ref}/[s]$	$t_{gF}/[s]$								R_{gF}
		number of cells:		8^3	9^3	10^3	11^3	12^3	13^3	14^3	15^3	
1	1728	1.0	15.27	67.14	54.71	48.93	46.40	82.89	76.42	71.51	66.82	3.04
2	1728	2.5	17.51			49.26	45.55	83.79				2.60
		number of cells:			20^3	21^3	22^3	23^3	24^3	25^3		
3	17576	1.0	40.79		112.2	109.3	106.8	104.0	101.7	182.8		2.49
4	17576	2.5	45.60				107.2	103.5	102.6	184.3		2.25

Table 12.2: Simulations comparing pr_04_2 ($r_{cut} = 2.5$) with gnemdFOAM.

no.	N_M	T	$t_{ref}/[s]$	$t_{gF}/[s]$								R_{gF}
		number of cells:		6^3	7^3	8^3	9^3	10^3	11^3	12^3	13^3	
5	1728	1.0	106.1	316.8	242.2	275.0	245.0	212.7	244.3	281.0	282.3	2.00
6	1728	2.5	115.5				247.7	211.9	242.8			1.83
		number of cells:			18^3	19^3	20^3	21^3	22^3	23^3		
7	17576	1.0	231.1		513.0	460.3	452.2	427.1	407.1	498.5		1.76
8	17576	2.5	251.3				452.0	427.9	408.3			1.62

to show the relative performance.

These comparative timing tests were carried out on PC with a 2.66GHz Intel® Core™2 Q6700 processor and both codes were compiled using gcc version 4.2.2 using the `-O3` optimisation option. For each simulation the average kinetic and total energy per molecule for the reference codes and gnemdFOAM agreed.

In these cases, gnemdFOAM is a factor of 1.6 to 3.0 slower than the reference codes which use a conventional neighbour list method. The performance difference between gnemdFOAM and the reference codes decreases with increasing system size, temperature and cut-off radius. It should be noted, however, that the test is not comparing codes written with the same intention: gnemdFOAM calculates intermolecular potentials via a generalised run-time potential selection mechanism that allows any number of species to be simulated. The reference codes are written and compiled specifically for their application; only a single species simulation is possible. Additionally, the particle tracking system in gnemdFOAM is more sophisticated and costly because it is required for complex geometries.

The dependence of the speed of calculation of the reference codes on system size and temperature is consistent with reduced neighbour-list lifetimes, as noted in section 5.2.2 and appendix C. The speed of gnemdFOAM is not significantly

influenced by the system temperature because it uses a cell-only method. This is demonstrated by the agreement between the $T = 1.0$ and $T = 2.5$ results; they were similar enough that simulations other than around the fastest mesh resolution for the $T = 2.5$ cases were considered unnecessary.

There are significant 'jumps' and local minima in the results when varying the mesh resolution for a given cut-off radius, this is best seen in the results for case 5. The mesh comprises regular cubic cells and the resolution in each direction can change only by integer values. Certain steps in resolution cause a transition between a situation where the volumes of the cells in the DILs are almost all within the cut-off radius, to a situation where the outer layer of cells have little of their volume within the cut-off radius. The transition is sharp because all cell's DILs change at once because all cells are the same size. Irregular meshes, such as the tetrahedral meshes used above, will not exhibit this effect.

12.3 Discussion

The accuracy of using PFEE mesh searching to determine which cells in the mesh should interact has been demonstrated by simulating a cubic MD system represented by 21 different tetrahedral meshes, ranging from 7520 to 196847 cells. The results are the same, to within numerical round off error, to those produced by gnemd. The computational cost of intermolecular force calculation with AICA depends on the mesh used but scales favourably with increasing refinement of the mesh. In particular, having a significant number of empty cells does not add much computational overhead. The accuracy of exchanging intermolecular force data across periodic and interprocessor boundaries using referred molecules and cells has been demonstrated, with the 21 tetrahedral meshes producing the same results when simulated in serial as on 12 and 32 processors.

12.3.1 Choosing a mesh

The performance of the algorithm is highly dependent on the mesh it is applied to. When choosing a mesh to use for a simulation the following points must be considered, and in some cases numerically tested and traded-off against one another:

- The time required to build cell interaction lists depends on the intermolec-

ular potential cut-off radius, r_{cut}, the number of cells and their shape, and the form of the mesh: a tetrahedral mesh will have a different balance of points : edges : cells compared to a hexahedral mesh, which will impact on the time taken for a PFEE search.

- Optimising the size of the mesh to minimise the time taken to run the simulation is complex and is case-specific.
- The implications of mesh refinement on the measurement of spatially resolved properties is discussed in section 10.1, but in general, the cell sizes are determined by whether property measurement or intermolecular force calculation requires finer mesh resolution.
- The mesh does not need to be uniform, it can have locally high resolution for making measurements in regions of interest but be coarser to improve computational speed if necessary in other regions. The 'tricks of the trade' used in CFD can also be applied, for example the mesh can be adaptively refined to track a region of interest.
- PP and PPGR are generally not good choices for building cell interactions unless the mesh in question comprises regular hexahedra because they can introduce significant errors to molecule force calculations.

Chapter 13

Case study: CAD-derived mixing channel

To demonstrate the capabilities of gnemdFOAM, a case study of flow in a complex, 3D nanochannel was simulated where the geometry was derived from a CAD (computer-aided design) model. The objective of the study was not primarily to analyse the fluid dynamics in the channel, rather to demonstrate what it is possible to simulate. A three inlet, one outlet microscale mixing channel from reference [177] was chosen as a guide for the geometry, see figure 13.1, and a reduced scale version was created.

13.1 Geometry definition

The process of creating the geometry is:

1. The system was drawn in Pro/ENGINEER®, a commercial CAD tool. A sketch is created defining the outline of the shape, see figure 13.2. Note that constraints are placed on most of the dimensions so that they may only be one of a small set of parameters, allowing the size of the mixing section to be easily changed. Adjusting L_2 adjusts the width of the mixing section. All dimensions are in MD reduced units, where $\sigma_r = 0.34nm$. This sketch is extruded to form a solid, hollowed out to leave a shell and the material at the inlets and outlets removed, see figure 13.3. The internal edges and corners are all rounded to make the geometry more realistic, as fabrication techniques are not able to make sharp channel shapes. This

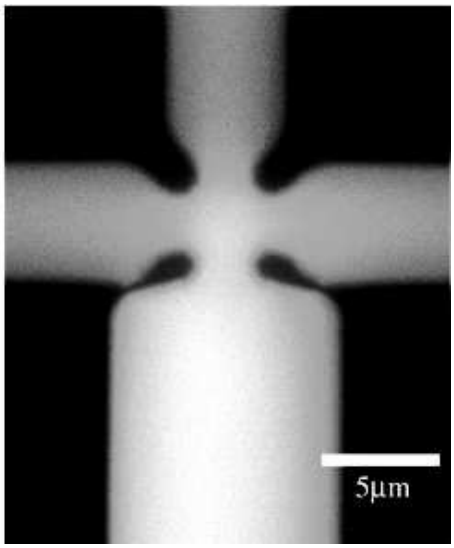


Figure 13.1: The geometry that the case study is based on, taken from [177]. There are three fluid inlets (narrower channels at the top of the image) and one outlet.

gives the volume of the wall regions with empty space remaining for the fluid section (figure 13.4).

2. The geometry is exported from Pro/ENGINEER® as a STEP file and imported into GAMBIT®, a commercial mesh generation tool, see figure 13.5.

3. A volume exists for the region that will form the solid walls and a region must be created for the volume containing the fluid. To do this, faces for the inlets and outlets are created from existing edges, as shown on figure 13.6.

4. These new faces are combined with the existing faces on the inside of the channel to create a volume for the fluid section. This is shown in figure 13.7 where the geometry involved only in the solid wall volume is not shown.

5. The fluid volume is split into 4 parts by creating box volumes surrounding each of the three inlets channels (see figure 13.8 for an example of this) and intersecting them with the fluid volume.

6. A reservoir volume is created at the entrance to each of the inlet channels and joined to the adjacent inlet channel volume, see figure 13.9.

7. Each volume is given a name to identify all of the cells that are created in it as being part of a zone for the generation of initial configurations of molecules.

8. Collections of geometrical faces are grouped together to form patches, so that all of the cell faces that lie on them will form patches in the mesh to which boundary conditions may be applied. The outlet, the faces of the molecule reservoirs that are external to the volumes and the remaining outer surface are defined as three separate patches.

9. The volumes are meshed by GAMBIT® using mostly automatic tetrahedral meshing, except in the molecule reservoirs where hexahedral cells were easy to create. A size function is applied at the mixing section to give finer cells around the mixing section. Creating meshes with tetrahedral cells is discouraged in continuum CFD, despite their ease of creation by automated tools, because they suffer from large interpolation errors compared to hexahedral cells. In this case the cells are required only to provide a space filling representation and to collect measurement data; no equations are solved using them. It will be seen, however, that collecting data in tetrahedral cells may produce noisy data.

10. The mesh is exported from GAMBIT® as a Fluent® mesh and imported into OpenFOAM using the fluentMeshToFoam utility with the 'writeZones' option. The mesh as imported is shown in figures 13.10, 13.11 and 13.12.

11. The boundary file in the OpenFOAM mesh directory is edited to assign the correct type to the patches created in the meshing process. The outlet patch and external surfaces except those of the molecule reservoirs are of type 'patch,' meaning that a molecule will be deleted when it touches a face. The reservoir external surfaces are of type 'wall,' and molecules are specularly reflected from faces they impact on. The external faces are of type 'patch' because the crystal near them is tethered forming a solid, so flow cannot leave the domain. The wall molecules whose tether points are close enough to the edge of the domain such that their locus of oscillation crosses the patch will be deleted. This does not affect the flow in the system if the wall is thick enough and is preferable to wall molecules close to the edge of the domain bouncing rapidly against a solid wall.

At this stage the geometry definition is complete.

13.2 MD preprocessing and simulation

Three types of molecule are created and are named depending where on where in the geometry they are initially placed. This allows the mixing in the channel to be observed by measuring the relative abundance of each type. There are two liquid molecule types, separated into those that start in the centre of the channel (the outlet region and the aligned inlet) and those that start in the side inlets. Molecules that comprise the wall are also created as a separate species to allow the channel boundary to be identified. The molecules have the ids LJ_C, LJ_S and LJ_W, meaning Lennard-Jones, Centre, Side and Wall)

Four simulations were carried out in this geometry at two temperatures, $T = 1.0$ in reduced units, equivalent to 120K, and $T = 2.5 \equiv 300K$. One simulation at each temperature used the same intermolecular potential between each molecule and another simulation used potentials where the energy scale of the shifted force Lennard-Jones potential was altered to make the liquids less miscible. To do this the attractive well was made deeper between liquid molecules of the same type and shallower between molecules of different types. The initial configuration of the systems was otherwise identical.

The molecule reservoirs are enclosed by solid walls and flow is created in the system by making the outlet type 'patch,' so that a molecule crossing it is deleted. This effectively means that the system vents into a vacuum, although the fact that a molecule is deleted as soon as it touches the patch means that it is not allowed to evaporate into the vacuum naturally: it is 'pulled' out of the simulation.

The potentials dictionary for both cases is shown in figure 13.15. The wall molecules are tethered into place by a harmonic spring potential, the strength of which is shown in figure 13.15. The steps required to preprocess and run the MD simulation are:

1. The mesh is decomposed into 48 portions using the METIS library [141], which is used by OpenFOAM's decomposePar utility. A script 'boundaryToProcs' is used to make a copy of the boundary description of the undecomposed mesh in the directory for each processor's mesh portion. This file is used in the construction of patch segments (see appendix D).

2. The domain is filled with molecules in parallel using molConfig. The zone specifications are given in figure 13.14 and the zone names are shown in figures 13.10 and 13.11. The liquid zones (LJ_C and LJ_S) are filled with

a simple cubic lattice and the wall molecules (LJ_W) are tethered into an FCC lattice. Figure 13.13 shows the molecules that are created on one of the processors, and the mesh portion belonging to an adjacent processor.

3. The controlDict (control dictionary file, common to all OpenFOAM simulations) has the timestep set to 0.005, with a simulation end time of 500 (both in reduced units). An interval of 10 time units (2000 timesteps) is specified to write the configuration of the system to disk (from which it can be restarted if necessary) and acts as the averaging period for measuring the field values. The temperature, velocity, number density and mass density fields are measured for each species and as total values. The mass and mole fraction fields can be calculated during post-processing using the ratio of species to total density.

The molecule creation and simulation runs were carried out on 48 cores of a 100 core cluster, each core belonged to a dual core, 64bit, 2GHz AMD Opteron^{TM} chip. The interconnect was via gigabit ethernet. It took approximately 13 minutes to create the initial configuration of 1462512 molecules for each case. The number of molecules created in all simulations is the same because the density, orientation, structure and anchor of the lattices stays the same for each. In conventional MD codes which simulate simple cubic domains, the computational cost of initial molecule generation is negligible. In this case 13 minutes is still a very small proportion of the overall simulation time. The MD simulations took between 80 and 136 hours to run. Building all referred cells and interaction lists took approximately 11 minutes and 30183 liquid molecules were removed at zone boundaries due to high energy overlaps. The time taken to solve varies partly due to the differing rate of outflow of molecules from the system (faster flow leaving fewer molecules to simulate in later timesteps) and also due to disk access and communications congestion caused by other jobs running on the cluster.

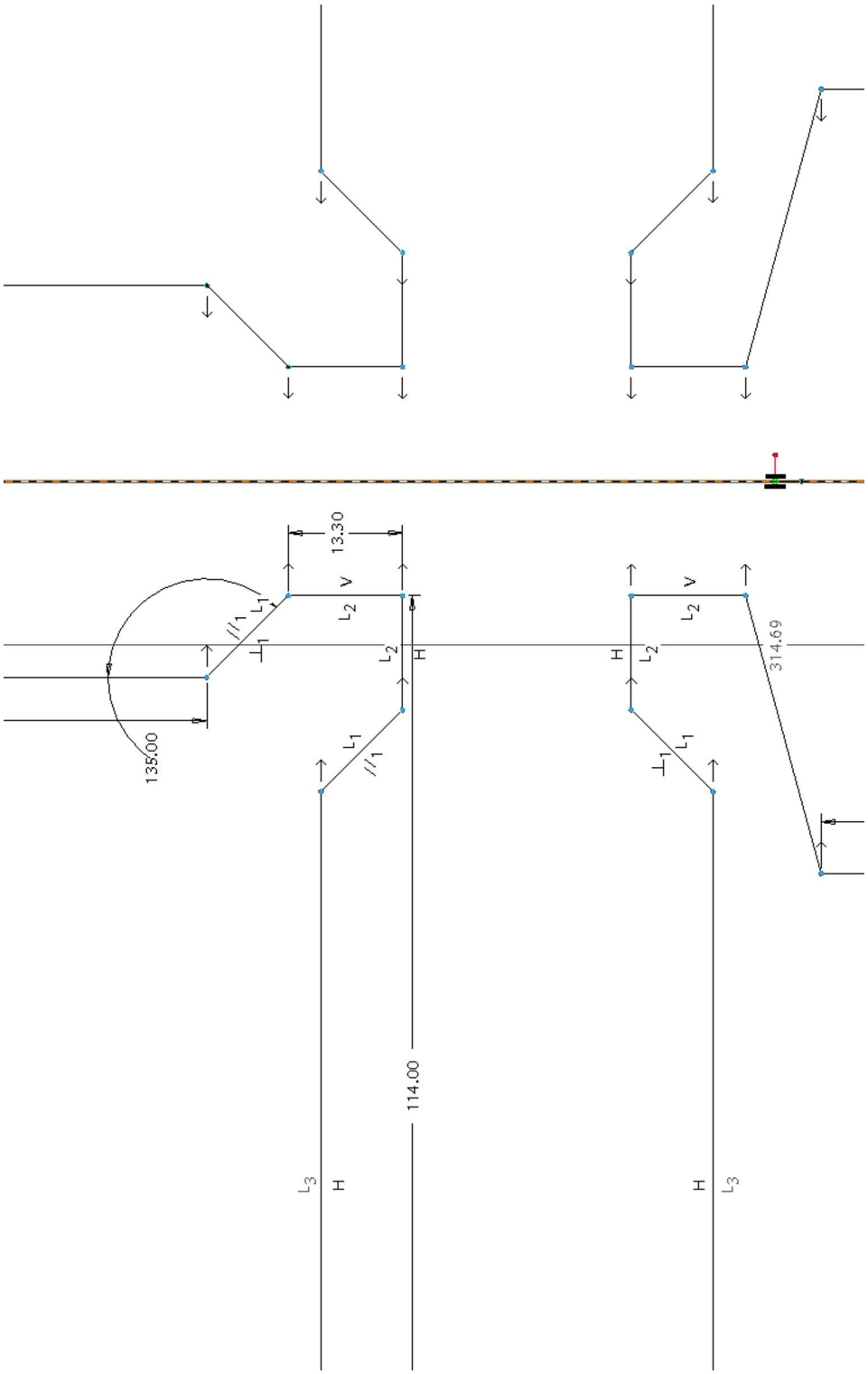


Figure 13.2: The Pro/ENGINEER® defining sketch of the mixing channel case study. Note that the right-hand-side is a mirrored copy of the left-hand-side, and that the mixing section dimensions are determined by parameter L_2. Therefore, changing this length (currently 13.3 reduced units) changes the whole mixing section as one entity.

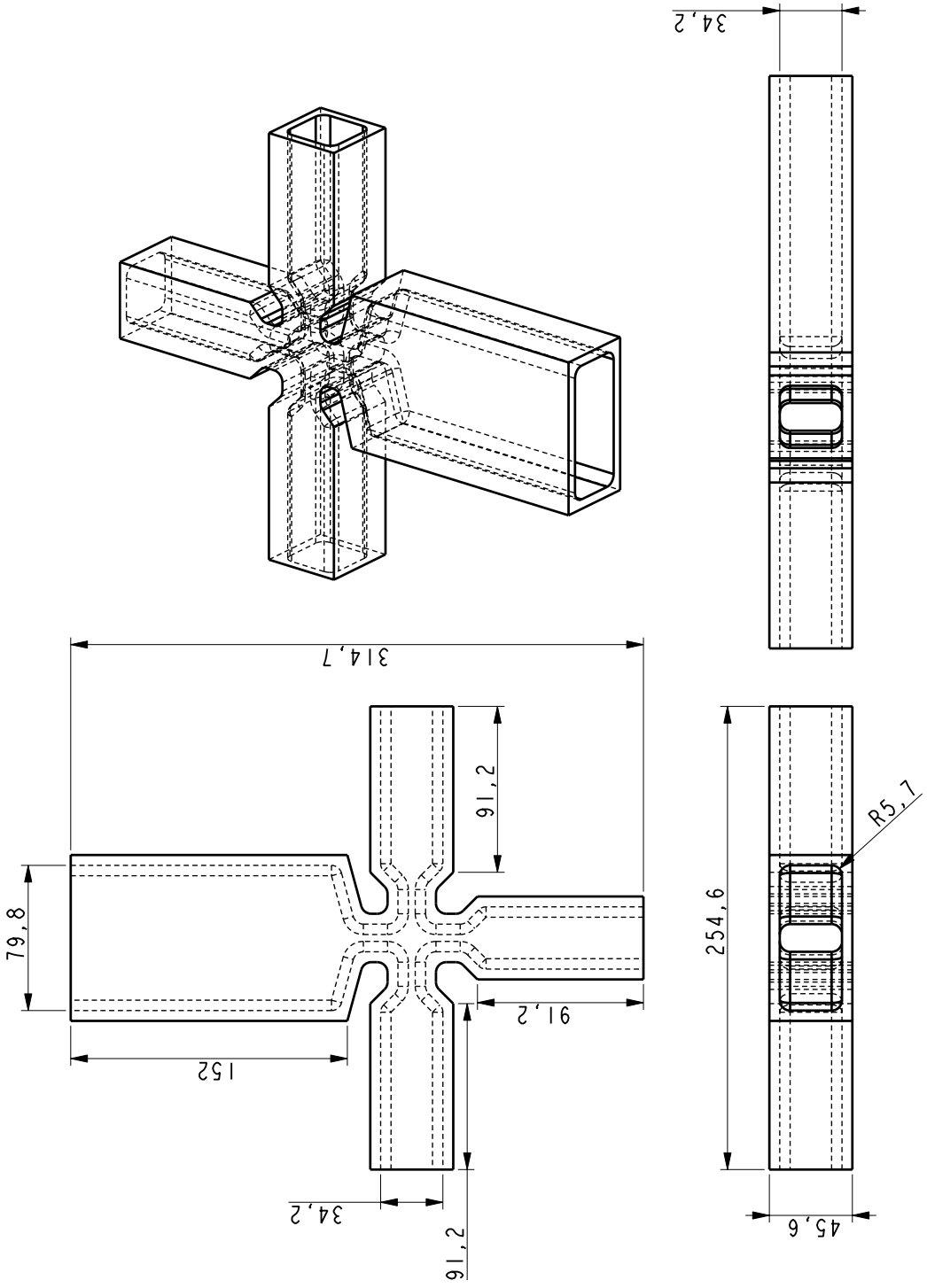


Figure 13.3: The overall dimensions, shown in reduced units where $\sigma_r = 0.34nm$, so the overall length of $314.7 = 107nm$.

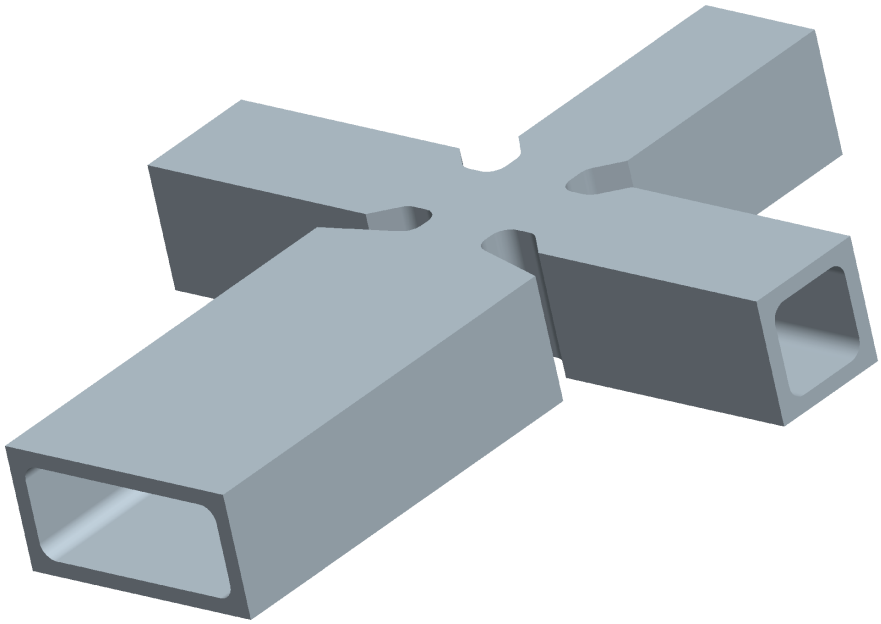


Figure 13.4: 3D view of the volume created in Pro/ENGINEER® to define the wall region of the channel.

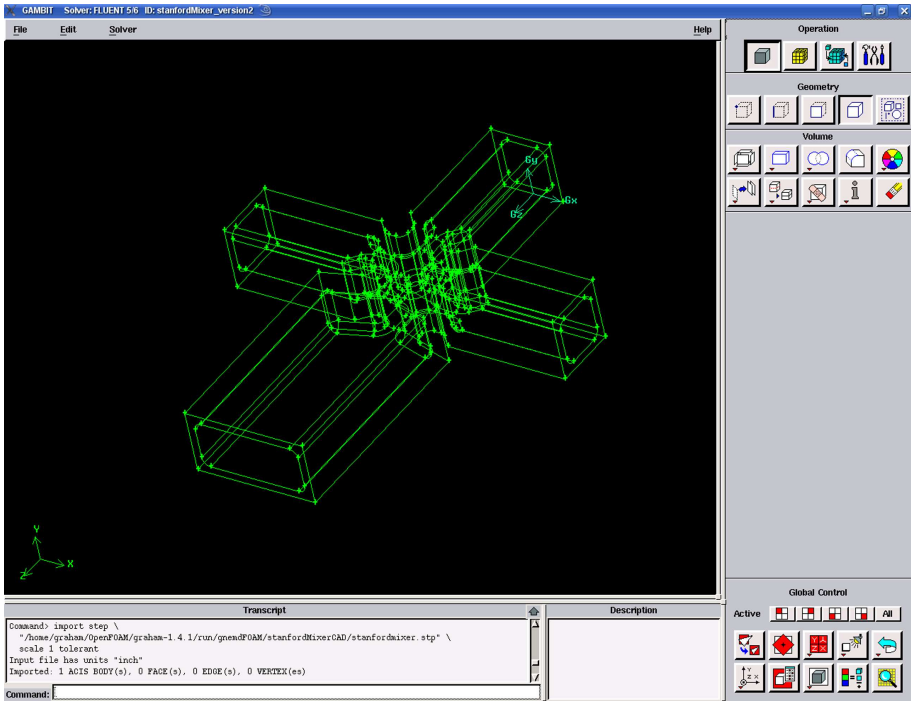


Figure 13.5: The geometry of the Pro/ENGINEER® model imported into GAMBIT® as vertices, edges, faces and a volume to represent the wall region.

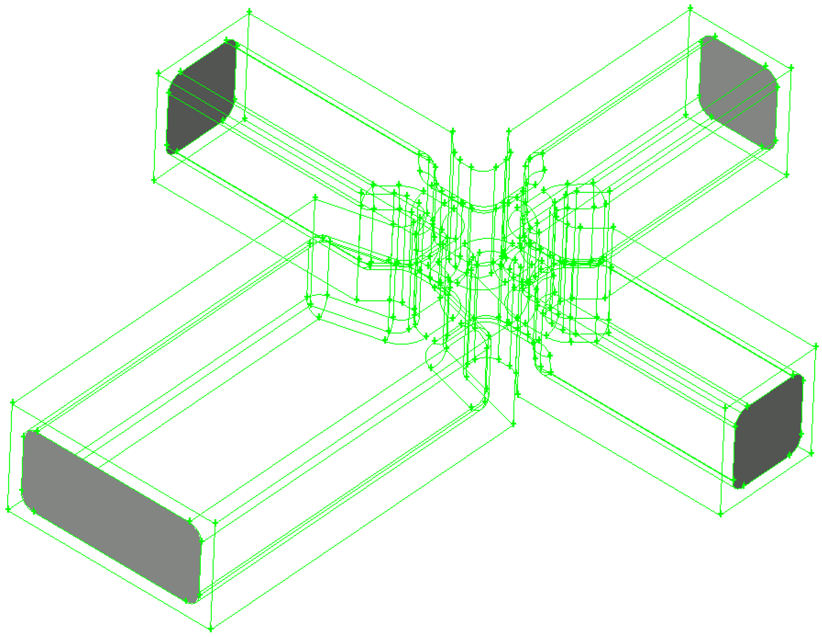


Figure 13.6: New faces are created using existing edges to define the inlet and exit faces from the fluid volume.

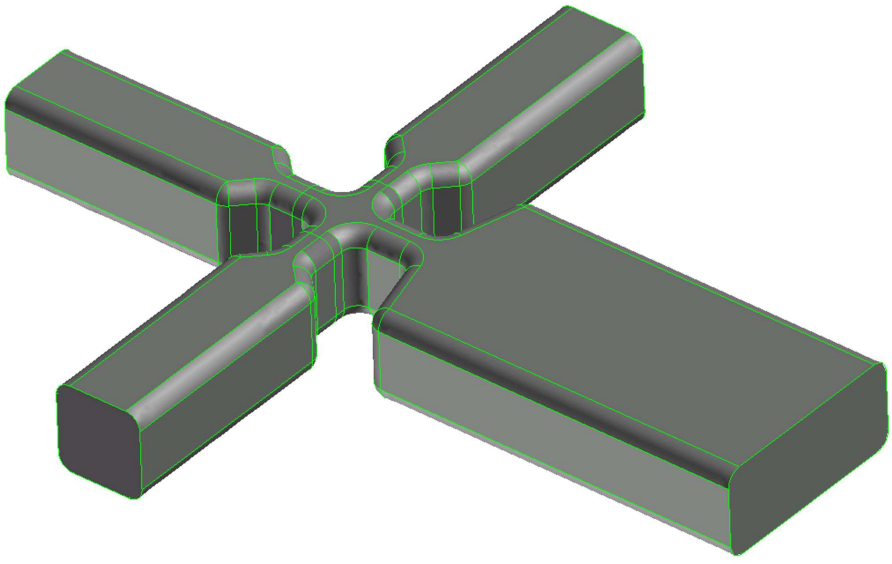


Figure 13.7: The newly created faces (see figure 13.6) are combined with the faces on the inside of the existing volume to define an internal volume for the fluid. The geometry not associated with this new volume is not shown

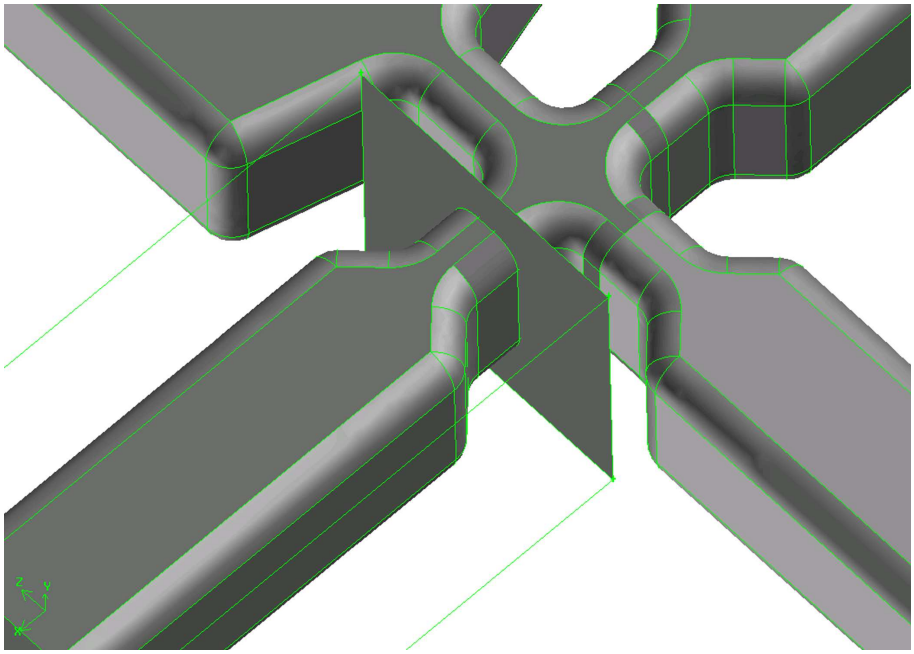


Figure 13.8: Rectangular volumes are created and used to split the fluid volume into four separate volumes so that the three inlets and the outlet region may be filled with molecules independently.

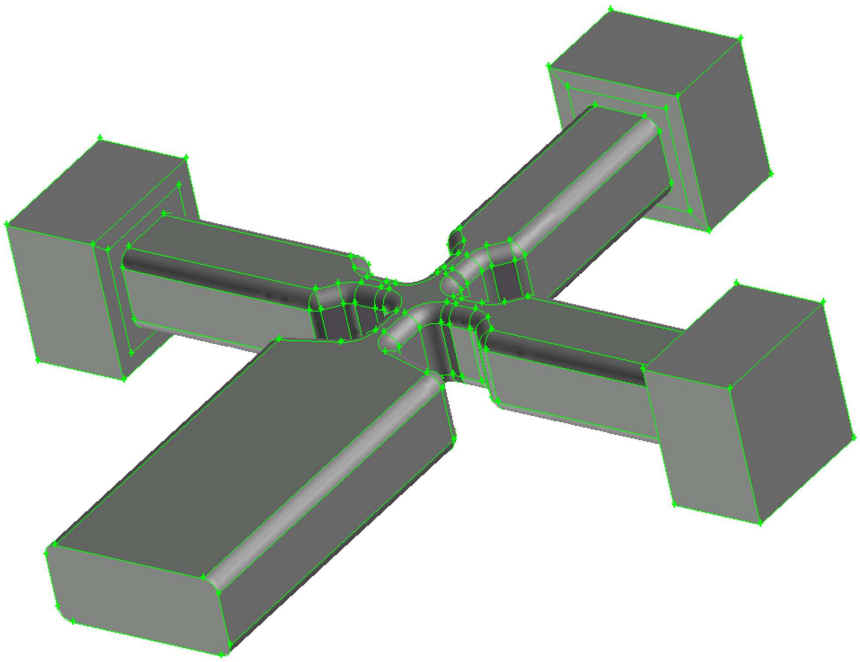


Figure 13.9: Molecule reservoirs are added to the fluid inlets.

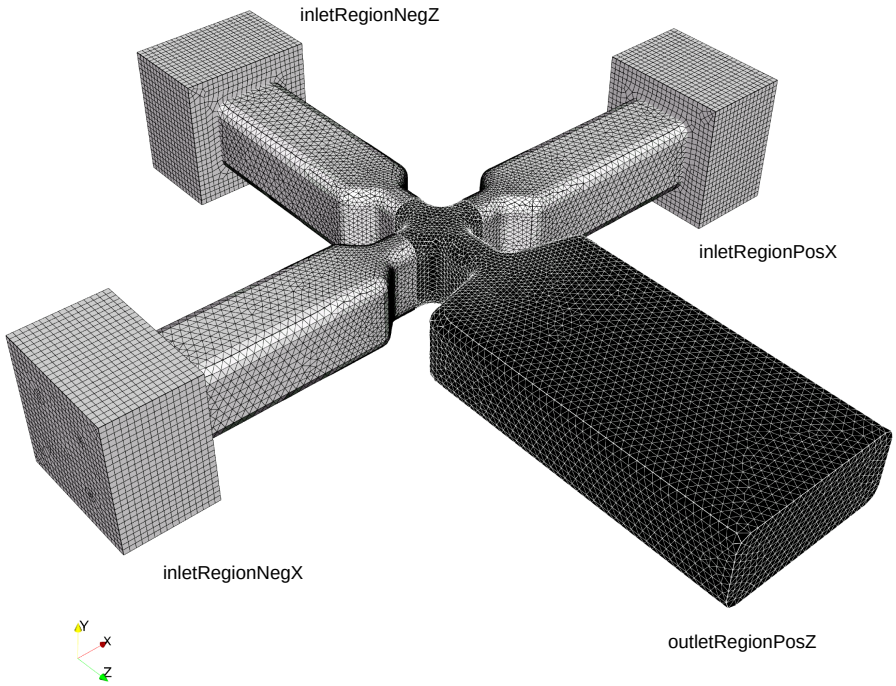


Figure 13.10: The mesh after being exported to OpenFOAM. The four liquid zones are shown.

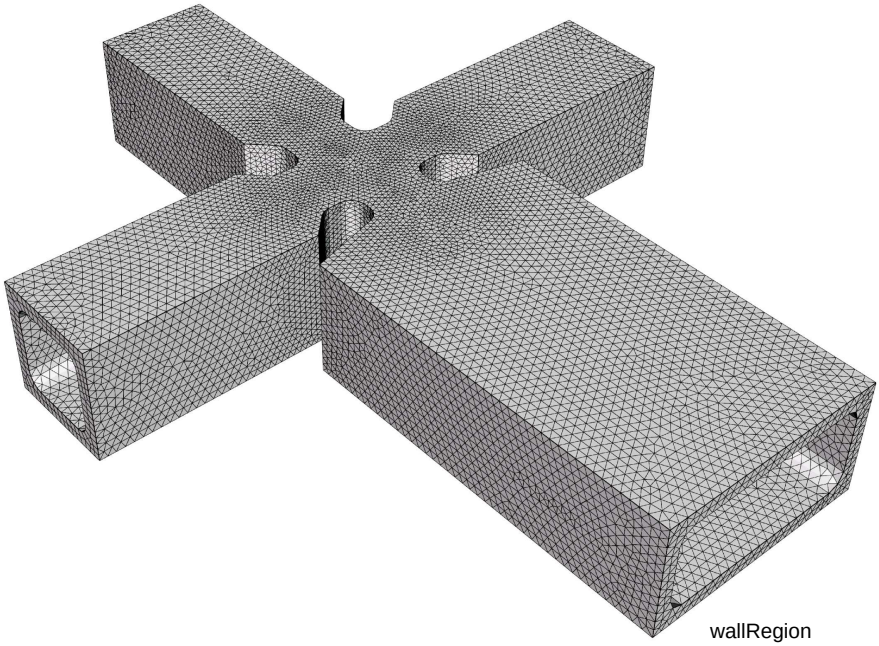


Figure 13.11: The mesh after being exported to OpenFOAM. The wall zone is shown.

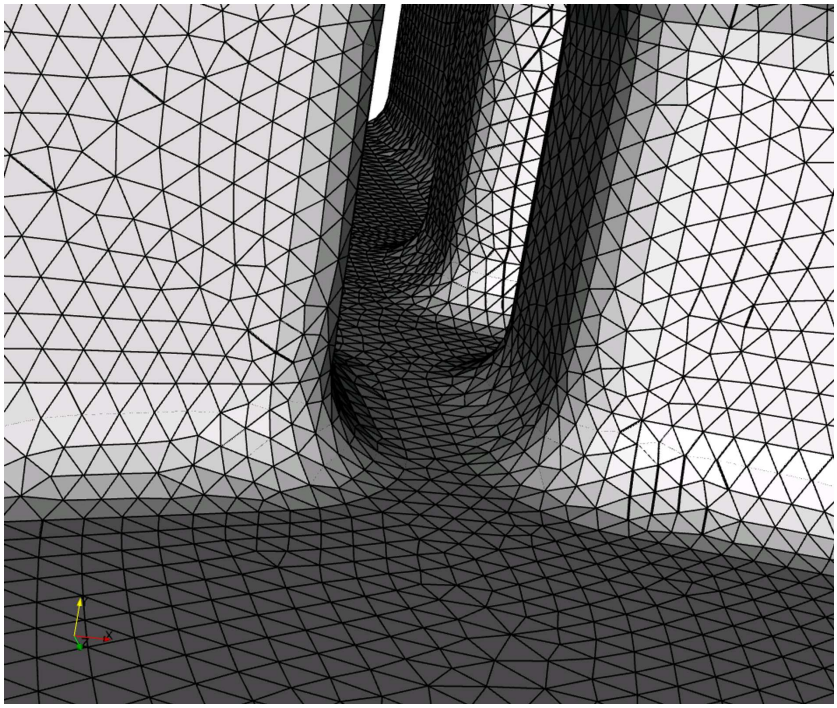


Figure 13.12: The internal space forming the liquid channel. Note the curved channel edges.

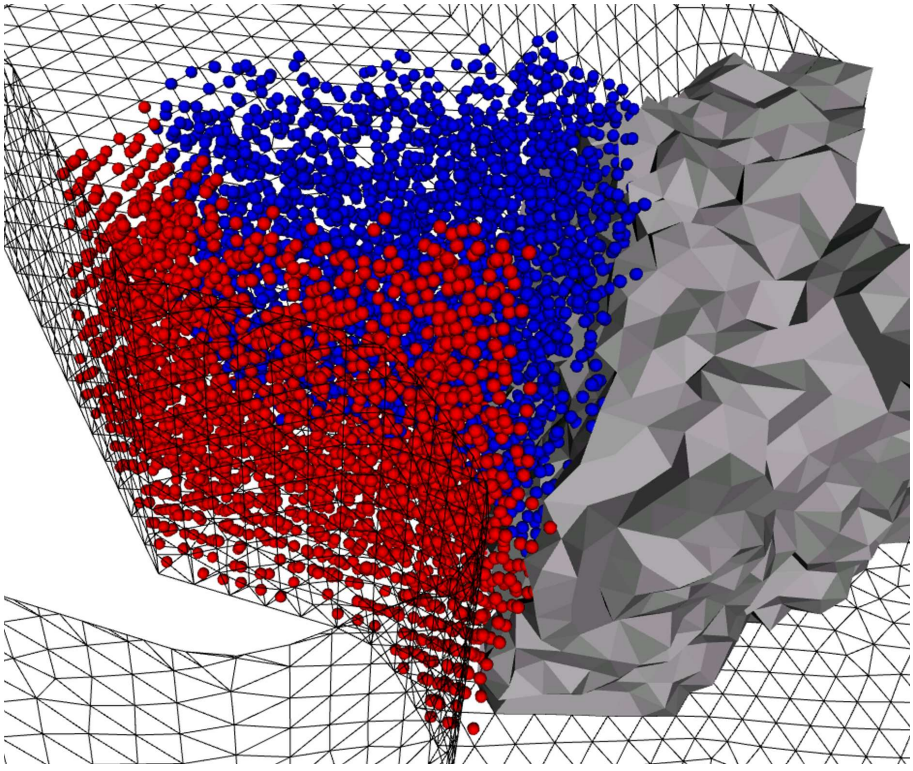


Figure 13.13: The molecules contained on one processor portion of mesh are shown coloured by id: blue for LJ_C and red for LJ_W. The mesh portion of an adjacent processor is also shown.

```
outletRegionPosZ
{
    density                 0.8;
    temperature             2.5; (or 1.0;)
    velocityDistribution    maxwellian;
    bulkVelocity            (0.0 0.0 0.0);
    id                      LJ_C;
    mass                    1.0;
    latticeStructure        SC;
    anchor                  (0.0 0.0 0.0);
    anchorSpecifies         corner;
    tethered                no;
    orientationAngles       (0 0 0);
}

inletRegionNegX
{
    density                 0.8;
    temperature             2.5; (or 1.0;)
    velocityDistribution    maxwellian;
    bulkVelocity            (0.0 0.0 0.0);
    id                      LJ_S;
    mass                    1.0;
    latticeStructure        SC;
    anchor                  (0.0 0.0 0.0);
    anchorSpecifies         corner;
    tethered                no;
    orientationAngles       (0 0 0);
}

wallRegion
{
    density                 0.85;
    temperature             2.5; (or 1.0;)
    velocityDistribution    maxwellian;
    bulkVelocity            (0.0 0.0 0.0);
    id                      LJ_W;
    mass                    1.0;
    latticeStructure        FCC;
    anchor                  (0.0 0.0 0.0);
    anchorSpecifies         corner;
    tethered                yes;
    orientationAngles       (0 0 0);
}
```

```
inletRegionNegZ
{
    density                 0.8;
    temperature             2.5; (or 1.0;)
    velocityDistribution    maxwellian;
    bulkVelocity            (0.0 0.0 0.0);
    id                      LJ_C;
    mass                    1.0;
    latticeStructure        SC;
    anchor                  (0.0 0.0 0.0);
    anchorSpecifies         corner;
    tethered                no;
    orientationAngles       (0 0 0);
}

inletRegionPosX
{
    density                 0.8;
    temperature             2.5; (or 1.0;)
    velocityDistribution    maxwellian;
    bulkVelocity            (0.0 0.0 0.0);
    id                      LJ_S;
    mass                    1.0;
    latticeStructure        SC;
    anchor                  (0.0 0.0 0.0);
    anchorSpecifies         corner;
    tethered                no;
    orientationAngles       (0 0 0);
}
```

Figure 13.14: Entries in the molConfigDict dictionary for the case study. The zone names are shown in figures 13.10 and 13.11

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// Removal order

removalOrder        3 (LJ_C LJ_S LJ_W);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// Pair potentials

pair
{
    LJ_C-LJ_C                                                   LJ_C-LJ_C
    {                                                           {
        potentialType   shiftedLennardJones;                        potentialType   shiftedLennardJones;
        sigma           1.0;                                        sigma           1.0;
        epsilon         1.0;                                        epsilon         1.25;
        rCut            2.5;                                        rCut            2.5;
    }                                                           }

    LJ_C-LJ_S                                                   LJ_C-LJ_S
    {                                                           {
        potentialType   shiftedLennardJones;                        potentialType   shiftedLennardJones;
        sigma           1.0;                                        sigma           1.0;
        epsilon         1.0;                                        epsilon         0.75;
        rCut            2.5;                                        rCut            2.5;
    }                                                           }

    LJ_C-LJ_W                                                   LJ_C-LJ_W
    {                                                           {
    potentialType   shiftedLennardJones;                        potentialType   shiftedLennardJones;
        sigma           1.0;                                        sigma           1.0;
        epsilon         1.0;                                        epsilon         1.0;
        rCut            2.5;                                        rCut            2.5;
    }                                                           }

    LJ_S-LJ_S                                                   LJ_S-LJ_S
    {                                                           {
        potentialType   shiftedLennardJones;                        potentialType   shiftedLennardJones;
        sigma           1.0;                                        sigma           1.0;
        epsilon         1.0;                                        epsilon         1.25;
        rCut            2.5;                                        rCut            2.5;
    }                                                           }

    LJ_S-LJ_W                                                   LJ_S-LJ_W
    {                                                           {
        potentialType   shiftedLennardJones;                        potentialType   shiftedLennardJones;
        sigma           1.0;                                        sigma           1.0;
        epsilon         1.0;                                        epsilon         1.0;
        rCut            2.5;                                        rCut            2.5;
    }                                                           }

    LJ_W-LJ_W                                                   LJ_W-LJ_W
    {                                                           {
        potentialType   shiftedLennardJones;                        potentialType   shiftedLennardJones;
        sigma           1.0;                                        sigma           1.0;
        epsilon         1.0;                                        epsilon         1.0;
        rCut            2.5;                                        rCut            2.5;
    }                                                           }
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// Tethering Potentials

tether
{
    LJ_W
    {
        potentialType   harmonicSpring;
        springConstant  150;
    }
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Figure 13.15: Entries in the potentials dictionary for the case study. In the pair potentials sub-dictionary the potential specification for the case where all potentials are identical (left) and when they are different (right) are both shown.

13.3 Results

The results from the simulations are shown in four series of figures. Each figure shows the total density, total velocity magnitude, mole fraction of the side species (LJ_S) and mole fraction of the centre species (LJ_C) for a particular averaging period. The four series are:

- figures 13.20 to 13.25 on pages 146 to 151: equal intermolecular potentials and $T = 2.5$ for timestamps 10, 50, 90, 130, 170 and 210;
- figures 13.26 to 13.31 on pages 152 to 157: different intermolecular potentials and $T = 2.5$ for timestamps 10, 50, 90, 130, 170 and 210;
- figures 13.32 to 13.37 on pages 158 to 163: equal intermolecular potentials and $T = 1.0$ for timestamps 10, 100, 200, 300, 400 and 500;
- figures 13.38 to 13.43 on pages 164 to 169: different intermolecular potentials and $T = 1.0$ for timestamps 10, 100, 200, 300, 400 and 500.

The timestamp for an averaging period refers to the data collected in the previous 10 units of time, so a timestamp of 170 represents the data collected between 160 and 170. The temperatures stated are the initial temperatures of the simulation, defined by molConfig. In each simulation the temperature drops because the system is depressurising and evaporating.

All results shown are from a cut through the middle of the domain along a plane with normal in the y direction (see figure 13.10 for the axes of the geometry). An example of the field variation in the other two planes is shown in figure 13.16.

The data is collected in cells, but can be displayed as an interpolated field by ParaView [144]. The two representations are compared in figure 13.17. Figure 13.18 shows one of the inlet sections where the molecule reservoir comprises hexahedral cells and the channel section comprises tetrahedrals. It would seem from this that tetrahedrals produce 'noisier' data, although the relative volume of the cells would need to be accounted for to examine this. Full flow field plots are rarely useful for quantitative analysis, rather for appreciating the flow dynamics and gaining insight into the processes that are occurring, so the interpolated fields will be shown. Any quantitative results would require the data to be reduced to single number or line graphs. The interpolated fields are necessary to calculate and display contours of the field variables.

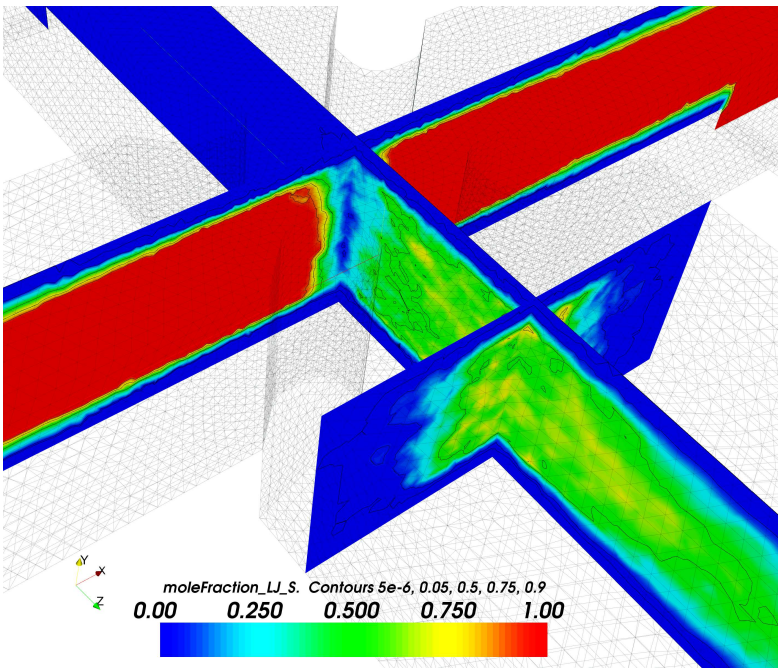


Figure 13.16: Cuts through the x and z normal planes of the domain.

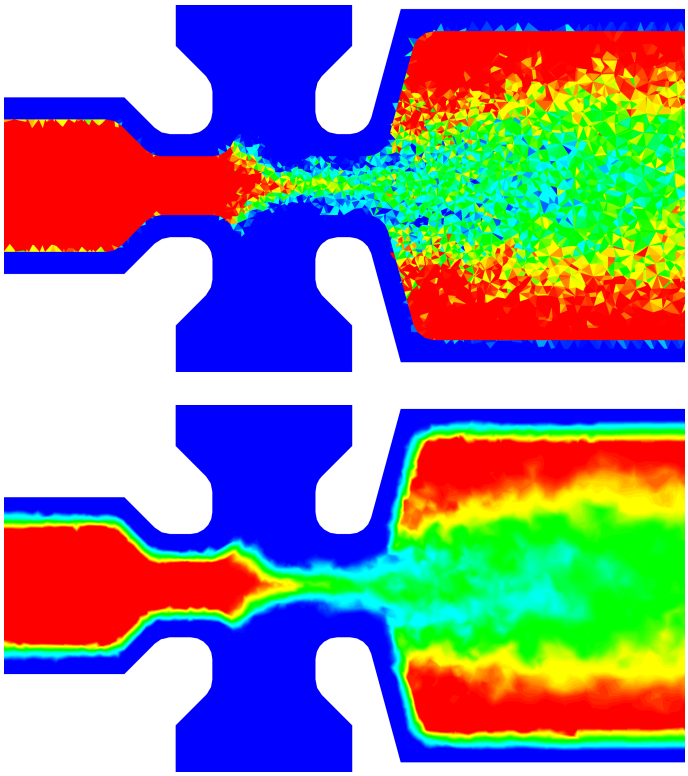


Figure 13.17: Comparing the raw cell data and the interpolated field. The raw cell data (above) is more difficult to appreciate in the mixing flow field but forms a sharp interface at the wall, the interpolated field (below) smears the interface at the wall but is easier to appreciate in the flow field.

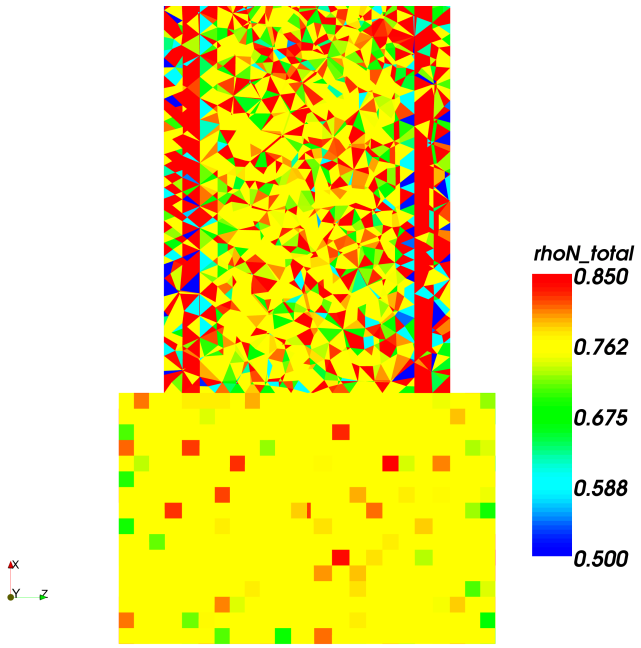


Figure 13.18: The cells in the molecule reservoirs are hexahedra, and those in the channel are tetrahedra. The hexahedral cell values seem to be less noisy.

There is a significant difference between the $T = 2.5$ and $T = 1.0$ simulations. In the $T = 2.5$ cases the outlet section rapidly depressurises and the flow 'chokes' at the throat of the mixer where very high velocities are observed; a velocity of 1.5 in reduced units is equivalent to $236m/s$. The side and centre species mix in a complex process because the state at the throat is close to the critical point for a Lennard-Jones fluid where $\epsilon^* = 1$ and $\sigma^* = 1$ [138]. Where the intermolecular potentials are different, the mixing of the two streams is not as complete; the side streams stay relatively separate from the centre. The 5×10^{-6} mole fraction contour shows the extent to which traces of one species have diffused into the other. Where the potentials are equal, the species diffuse into each other more readily.

In the $T = 1.0$ cases the flow is significantly slower and does not undergo a large decompression in the outlet section. The equal potential simulation forms a 'barrier' — an increase in density — at the exit. This could be caused when the outlet patch deletes a molecule and the molecules close to the exit that it was previously attractively interacting with recoiling due to the sudden removal of this downstream force. The equal potential fluids mix well and diffuse into each other, whereas in the different potential simulation the fluids stay relatively immiscible and do not diffuse into each other significantly. The different potential

simulation does not decompress as much, or flow as fast as the equal potential simulation.

The overall outflow can be appreciated by plotting the total number of molecules in the system as a function of time, see figure 13.19.

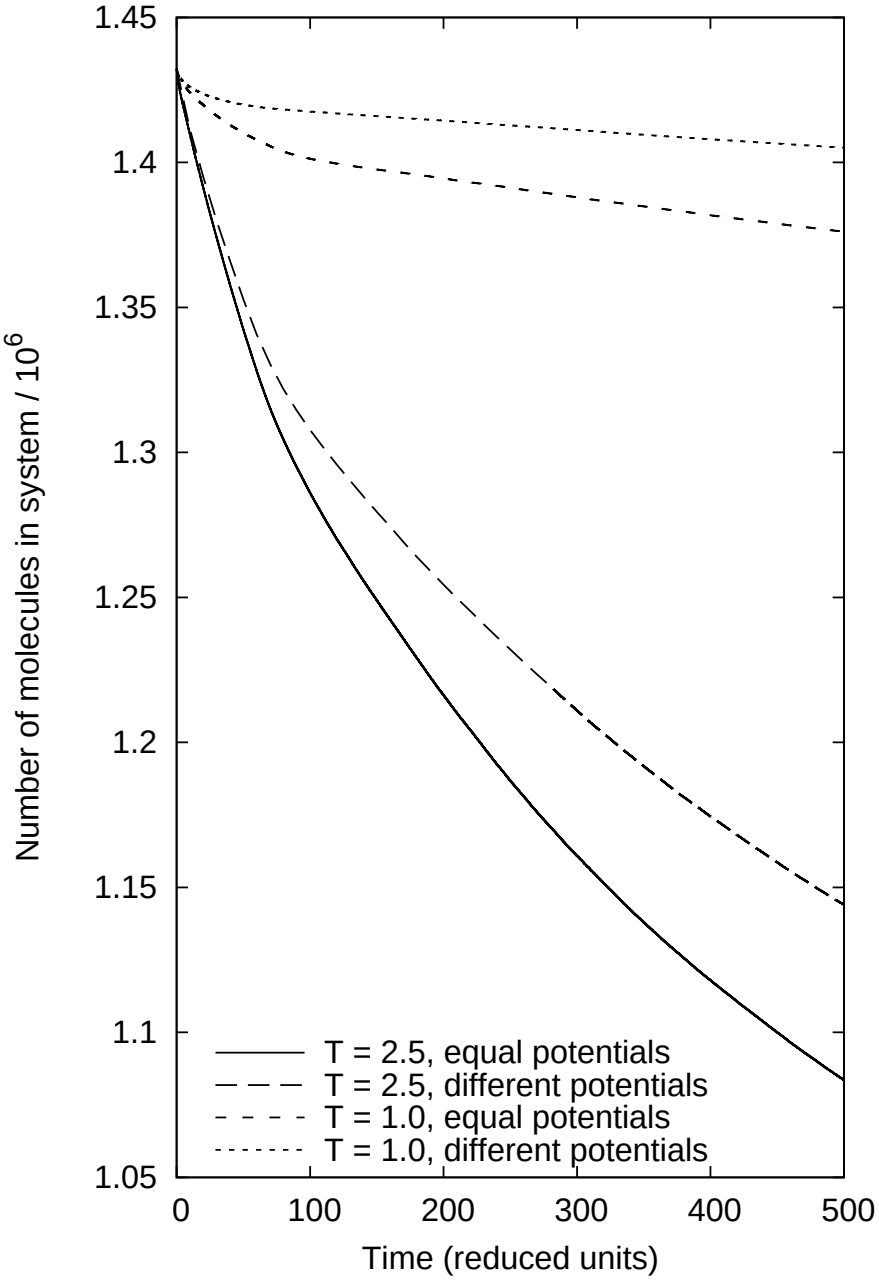


Figure 13.19: The total number of molecules in the system versus time. The initial decompression of the system can be seen in the steeper gradient sections. Reducing the temperature reduces the flowrate. The different intermolecular potential simulations flow more slowly than those where the potentials are equal.

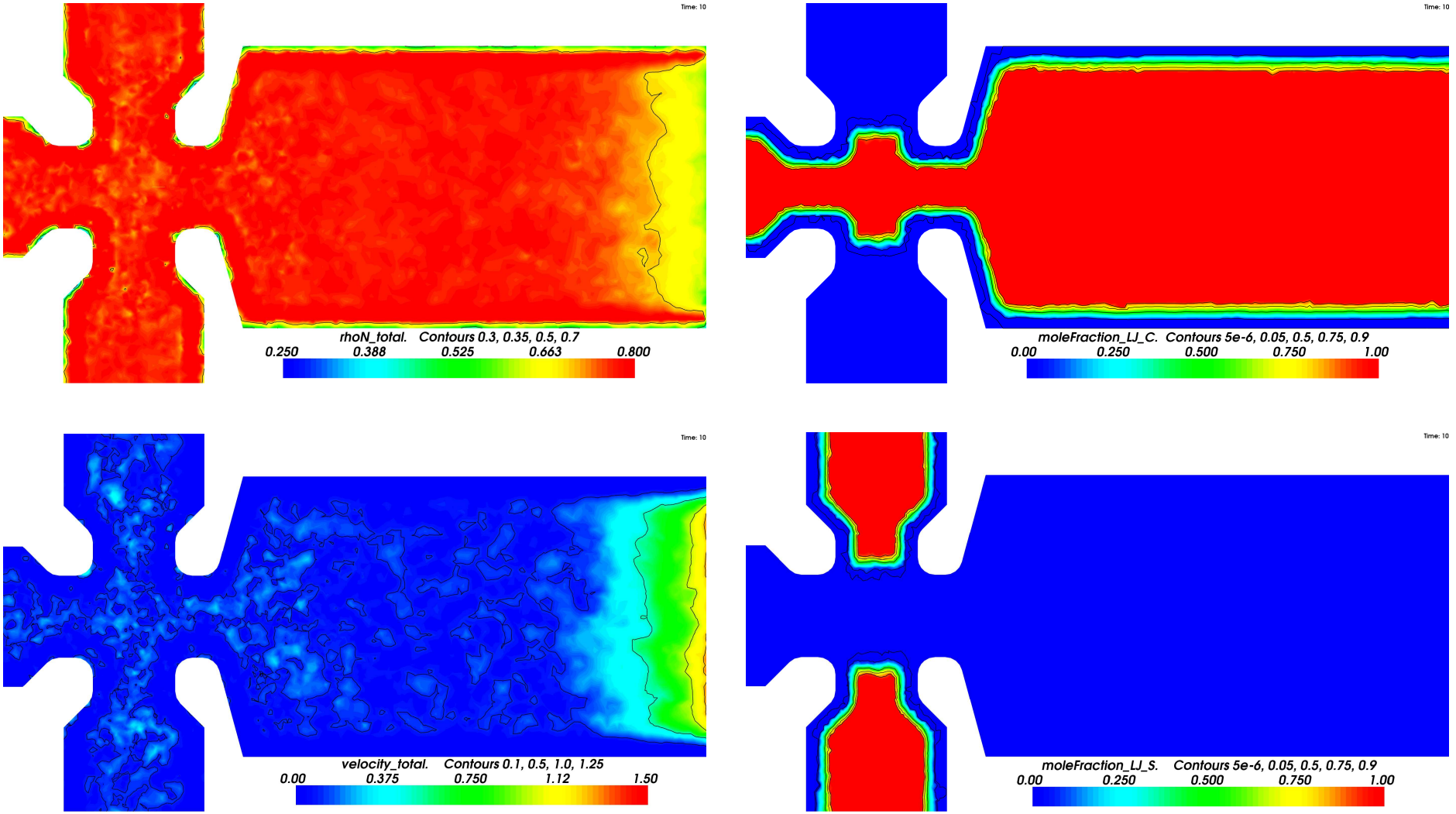


Figure 13.20: Equal potentials, $T = 2.5$, time $= 10$

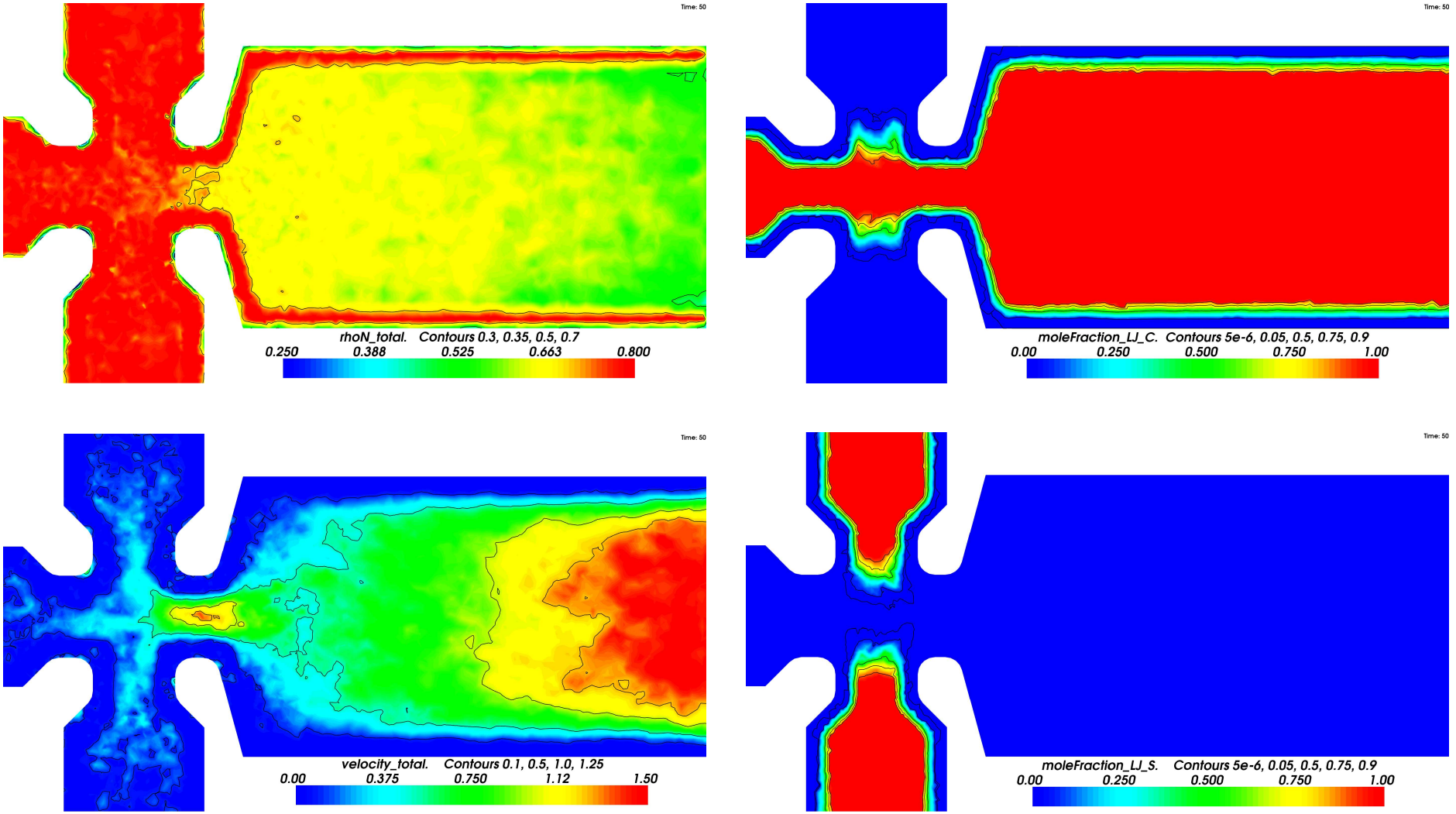


Figure 13.21: Equal potentials, $T = 2.5$, time $= 50$

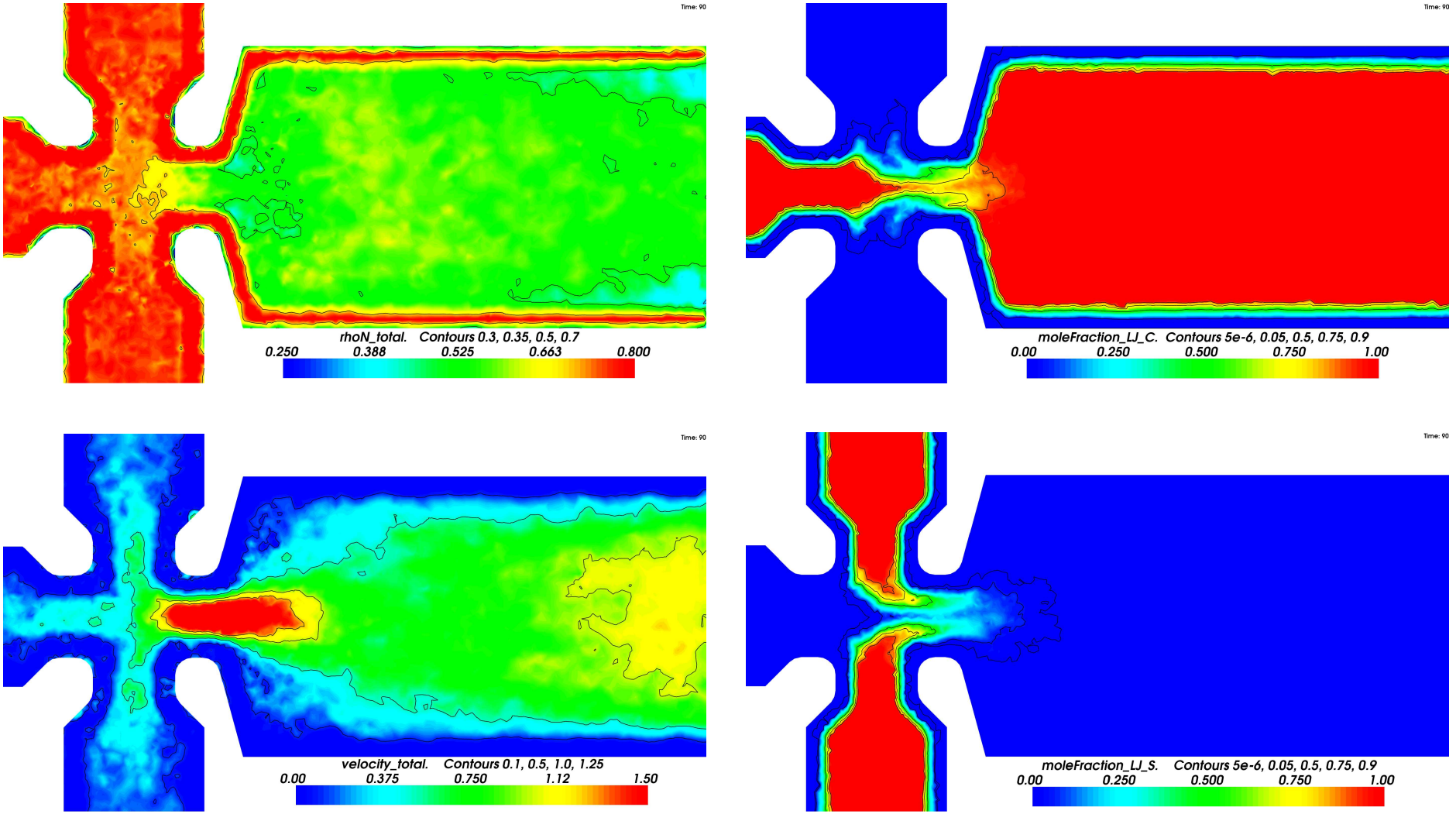


Figure 13.22: Equal potentials, $T = 2.5$, time $= 90$

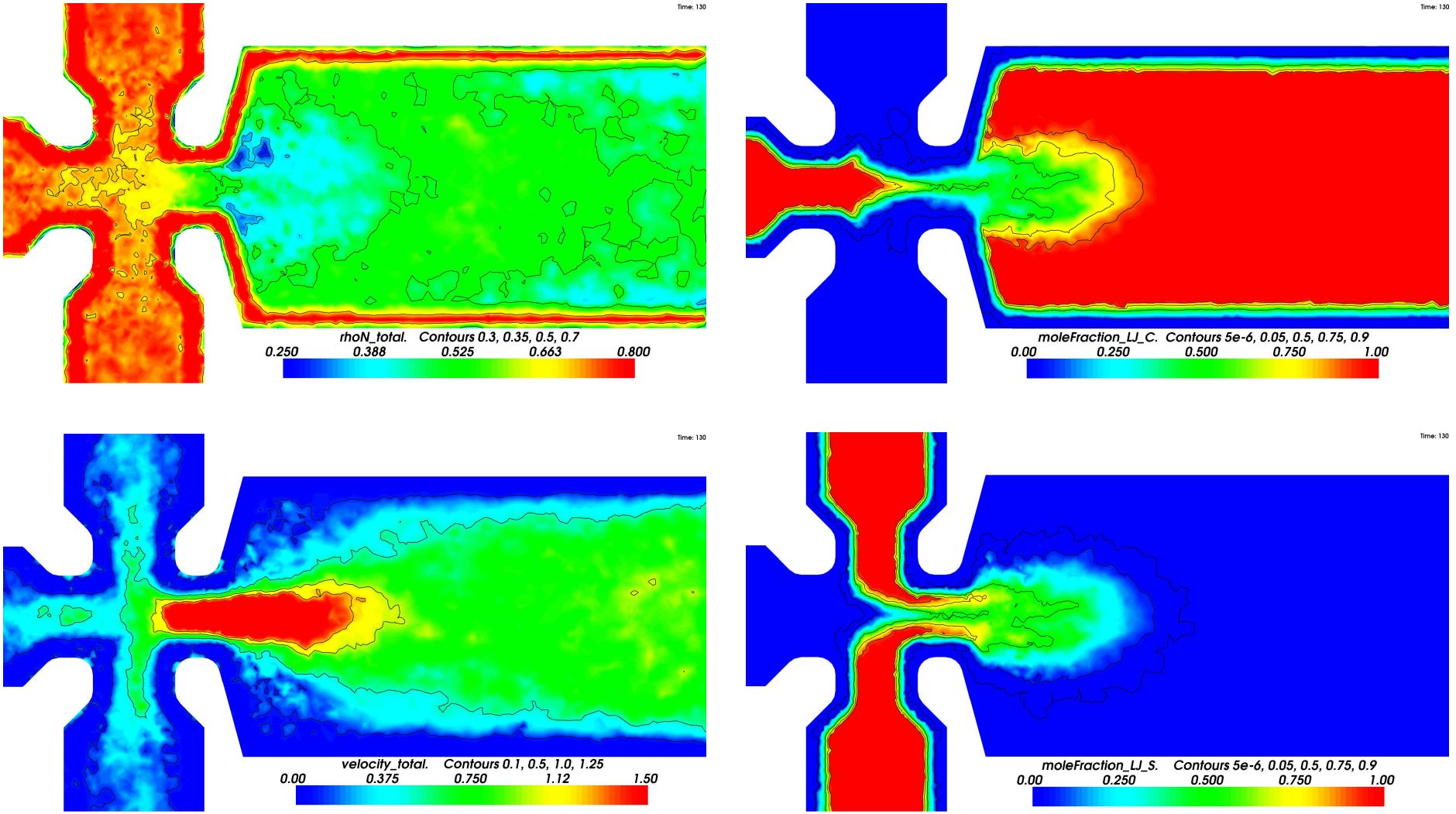


Figure 13.23: Equal potentials, $T = 2.5$, time $= 130$

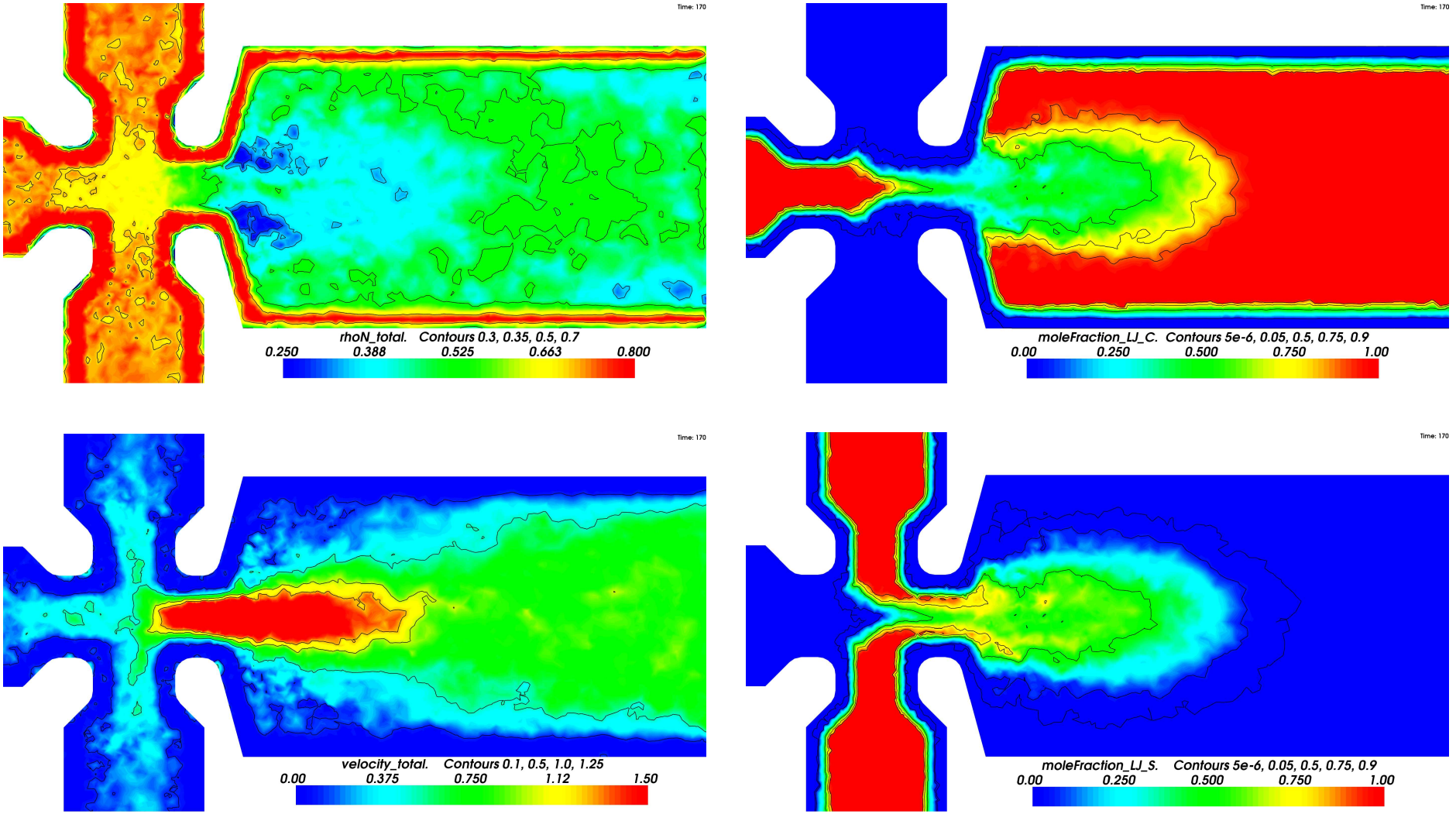


Figure 13.24: Equal potentials, $T = 2.5$, time $= 170$

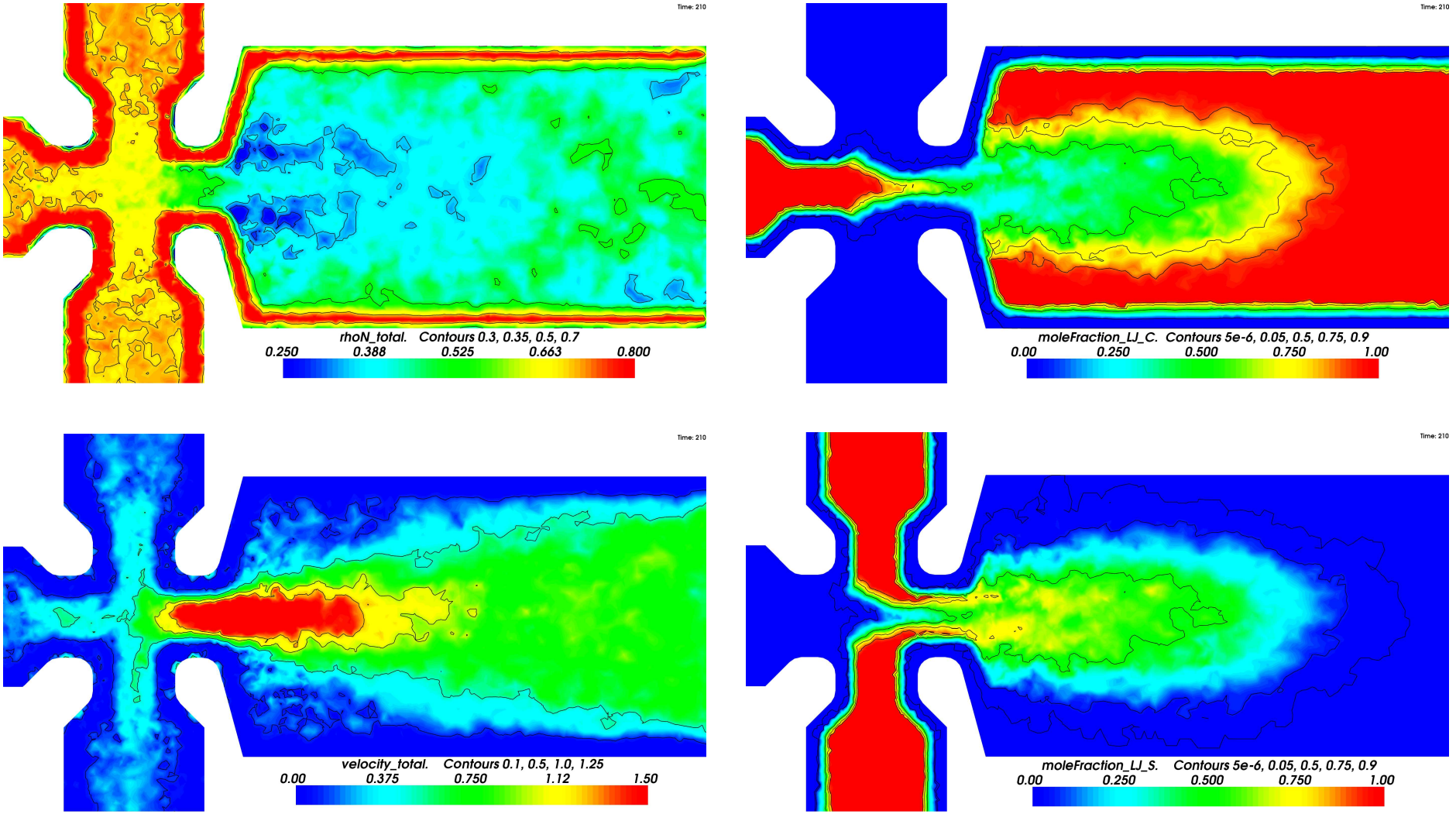


Figure 13.25: Equal potentials, $T = 2.5$, time $= 210$

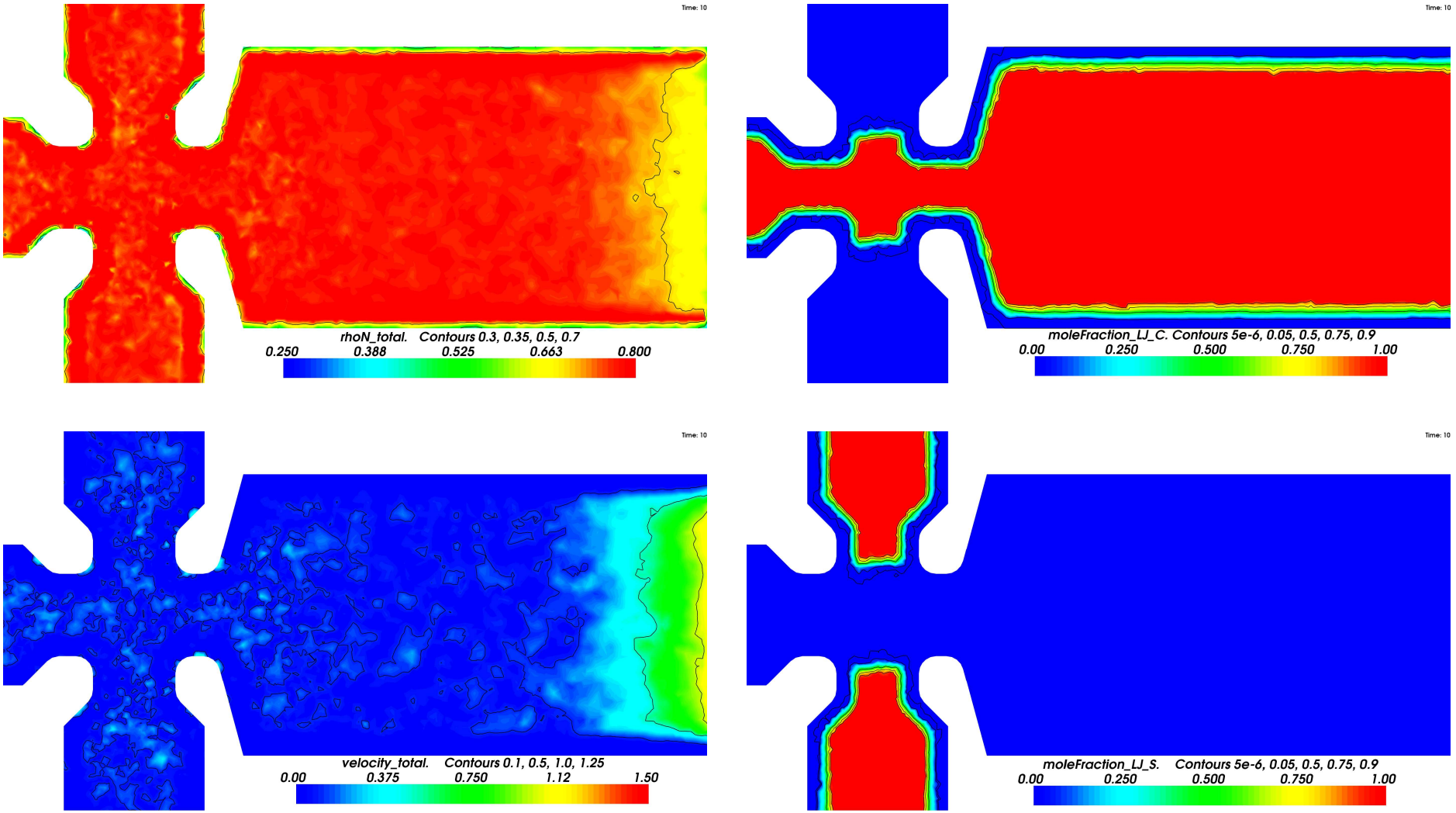


Figure 13.26: Different potentials, $T = 2.5$, time $= 10$

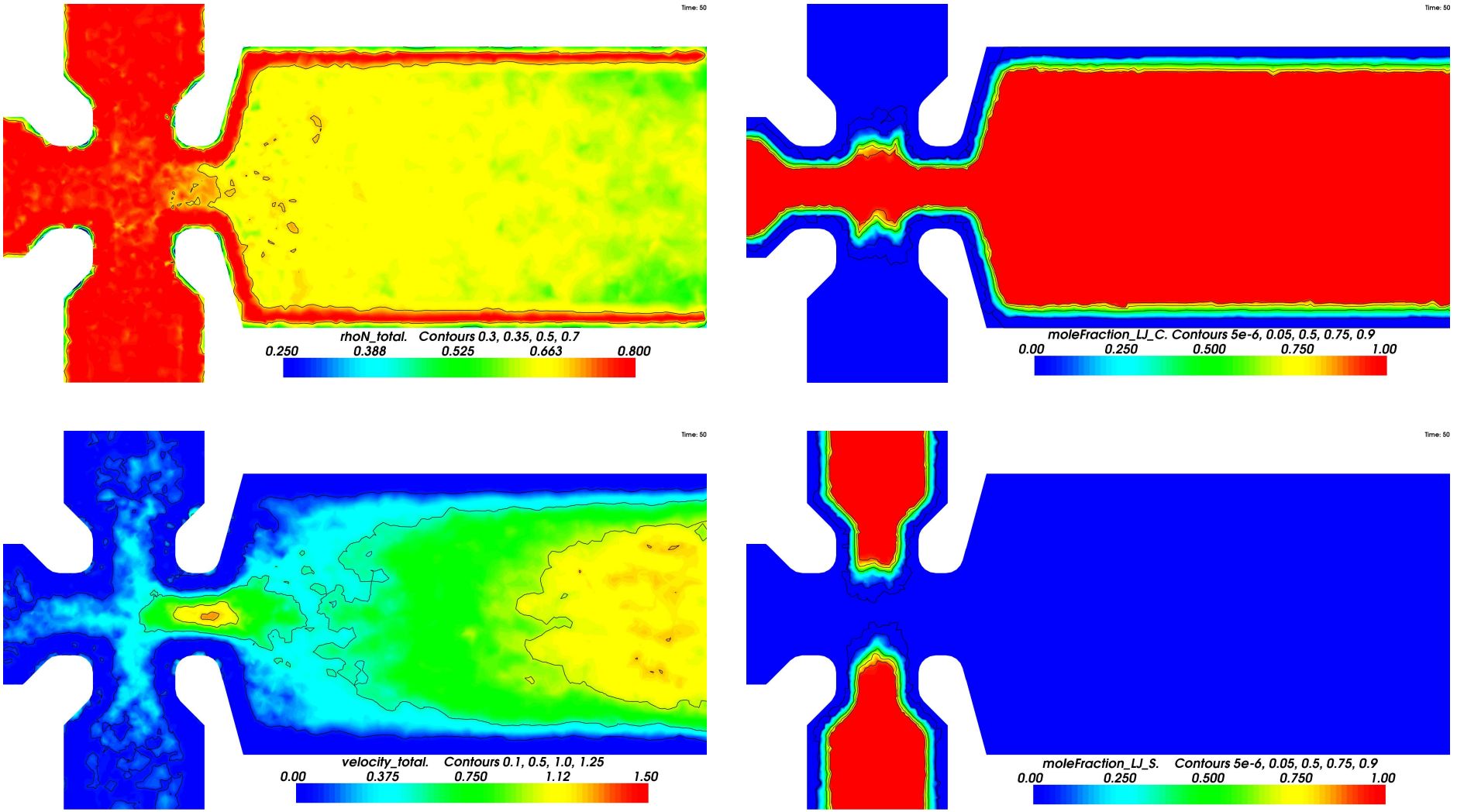


Figure 13.27: Different potentials, $T = 2.5$, time $= 50$

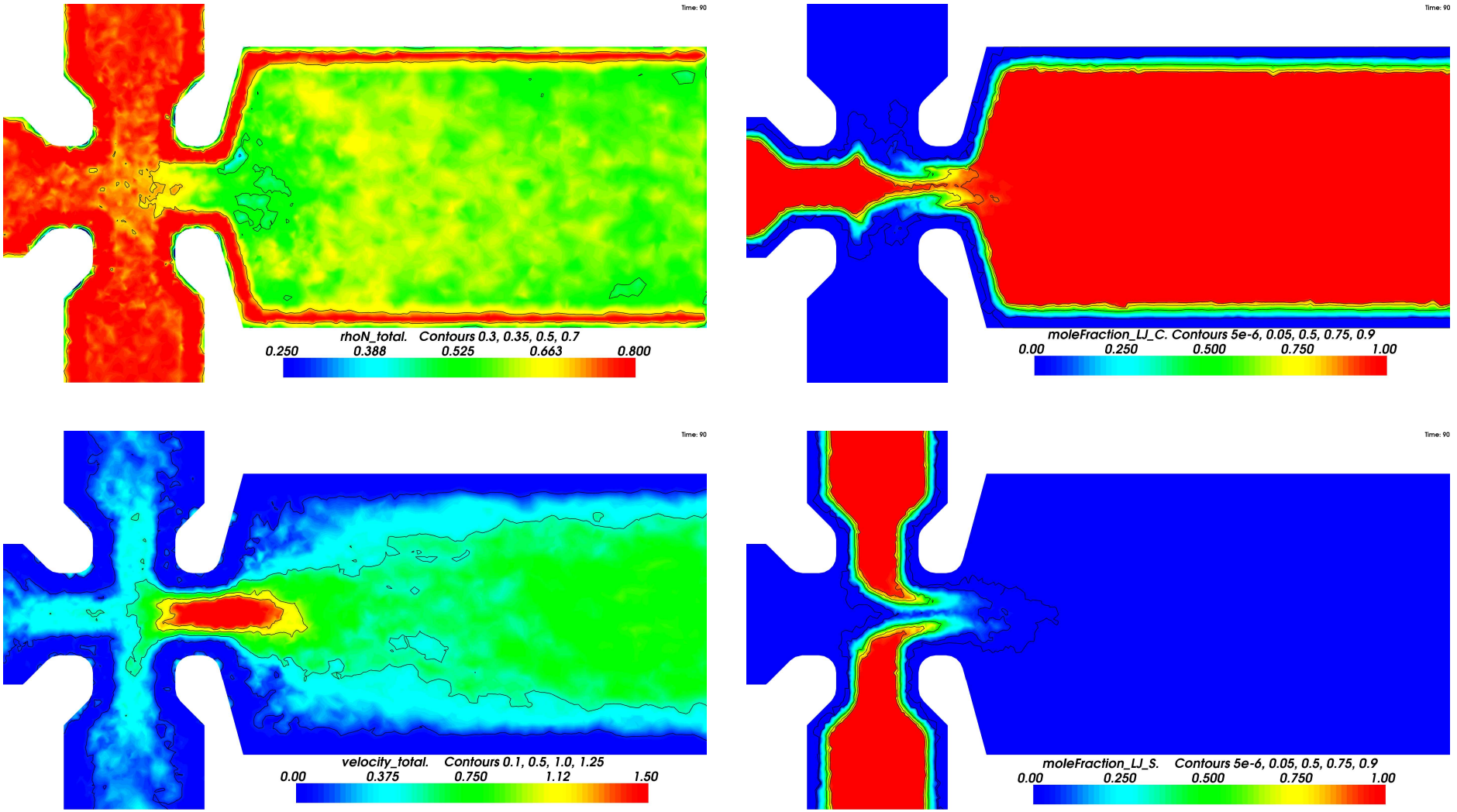


Figure 13.28: Different potentials, $T = 2.5$, time $= 90$

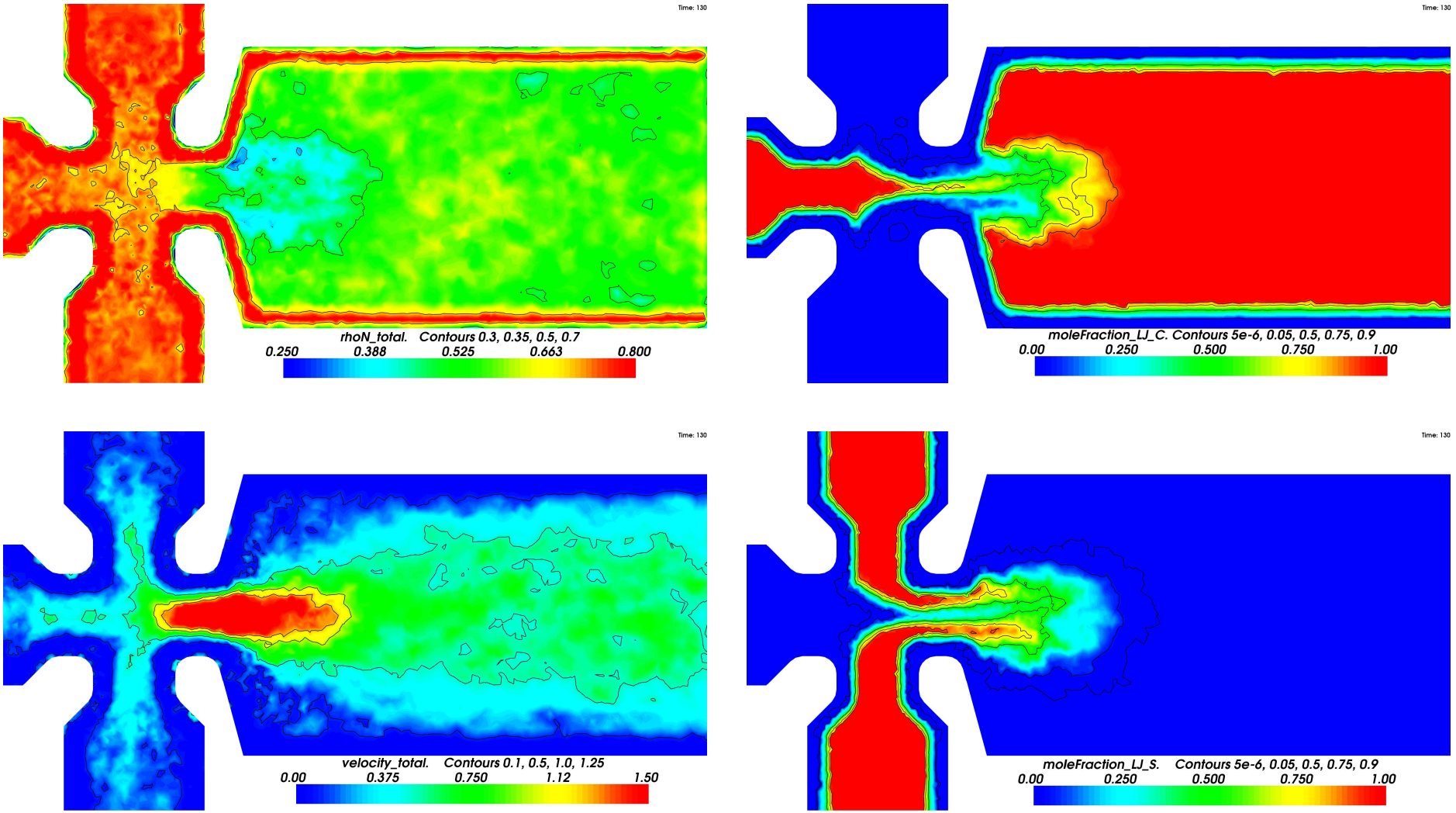


Figure 13.29: Different potentials, $T = 2.5$, time $= 130$

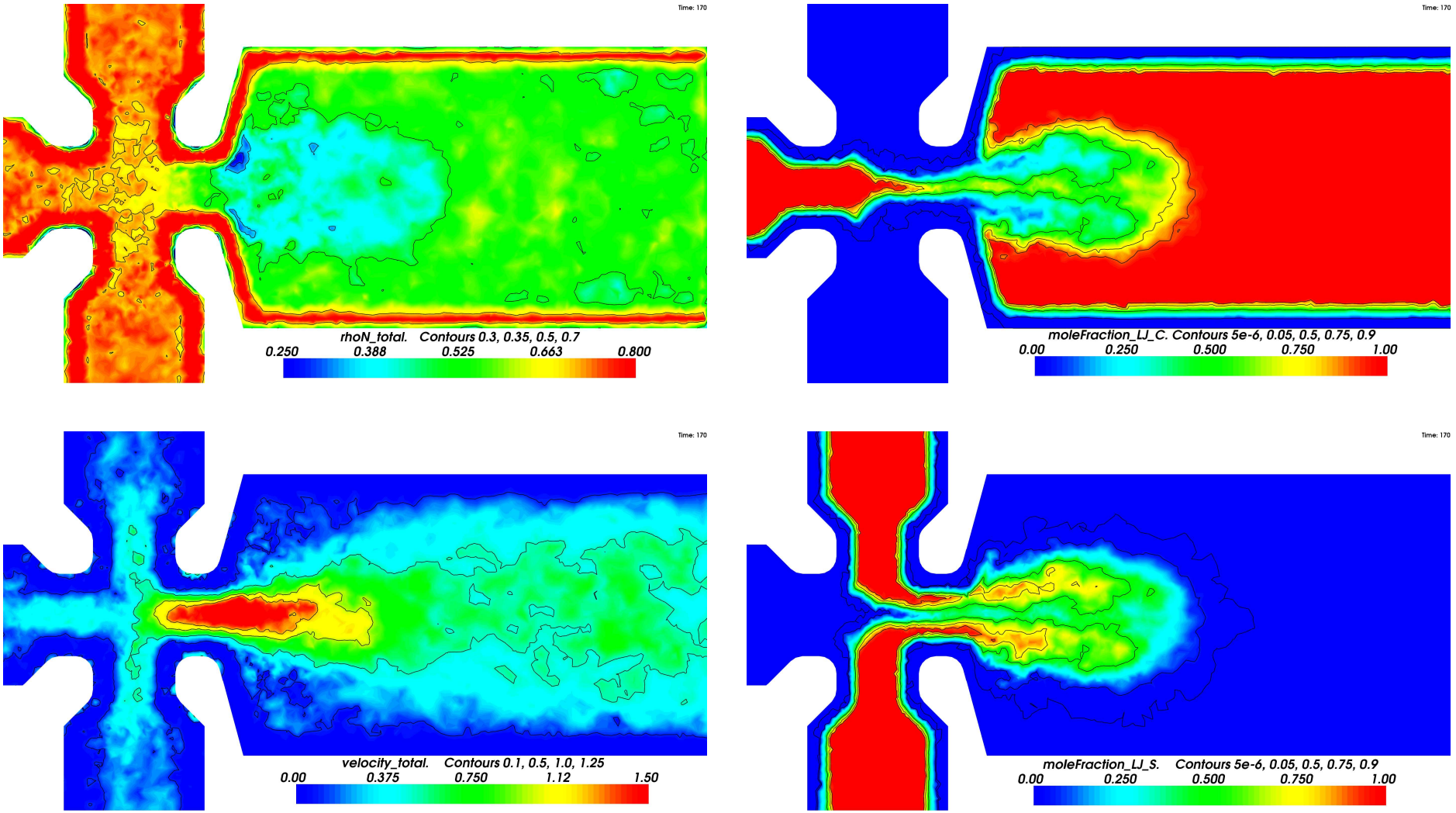


Figure 13.30: Different potentials, $T = 2.5$, time $= 170$

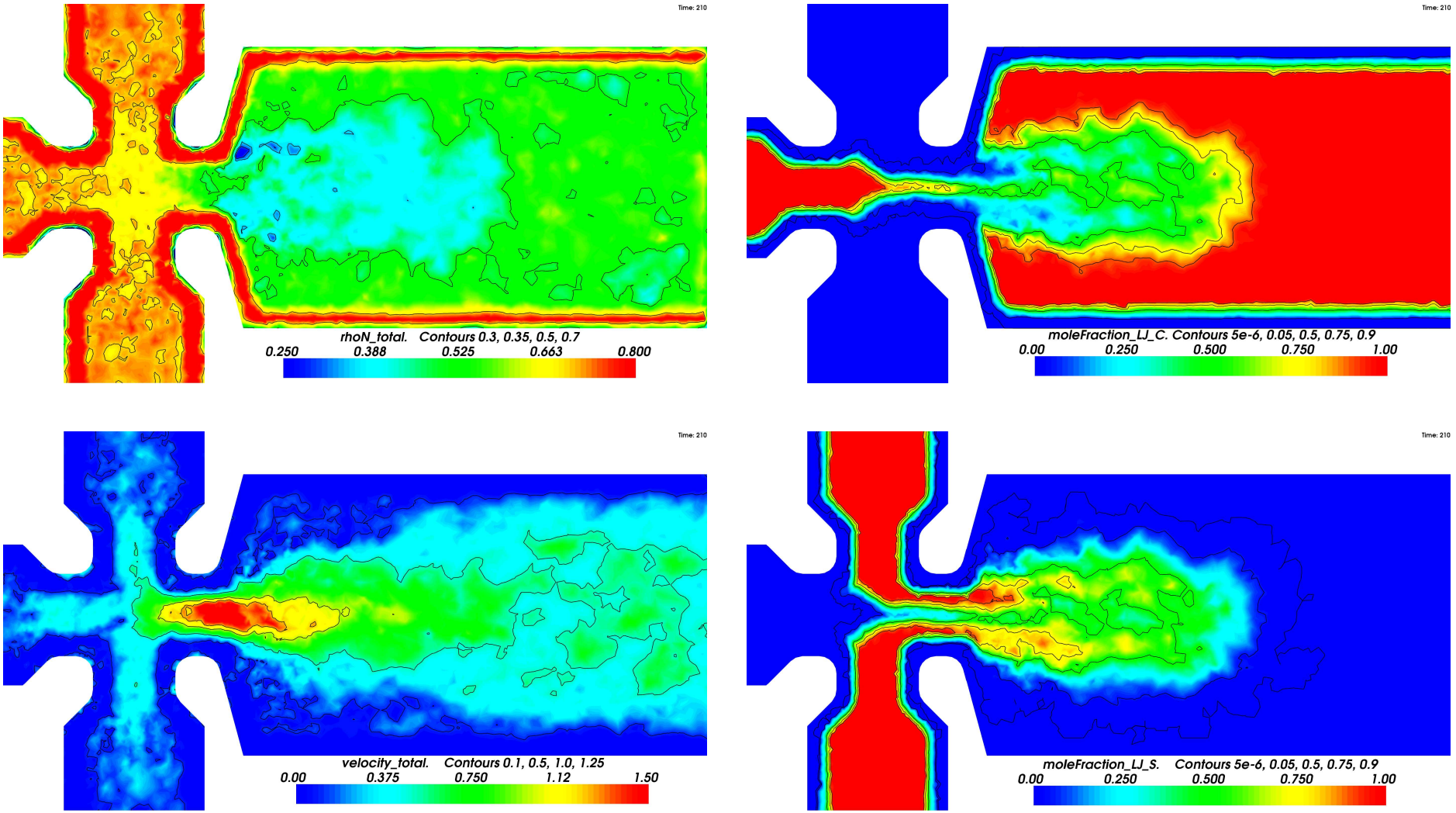


Figure 13.31: Different potentials, $T = 2.5$, time $= 210$

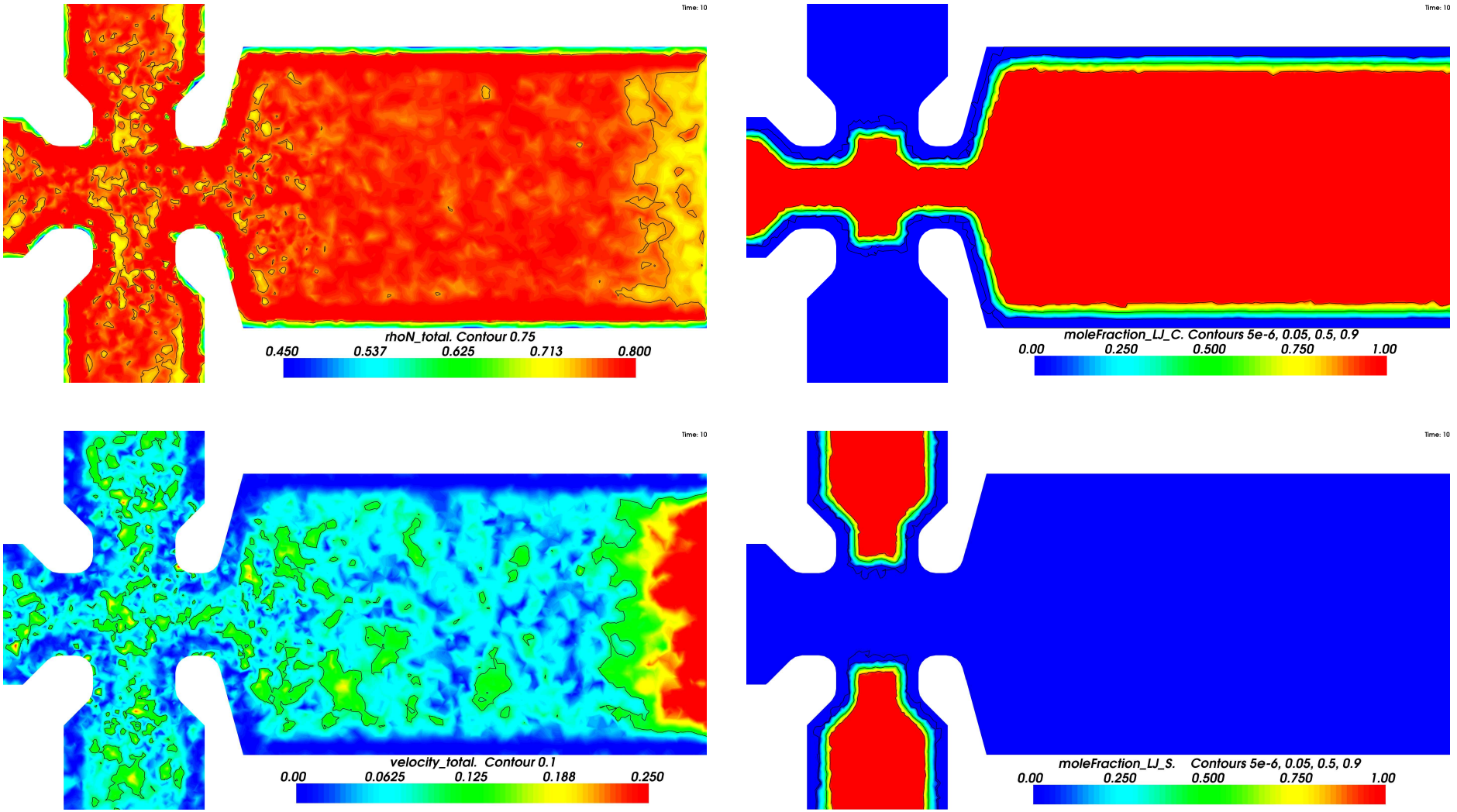


Figure 13.32: Equal potentials, $T = 1.0$, time $= 10$

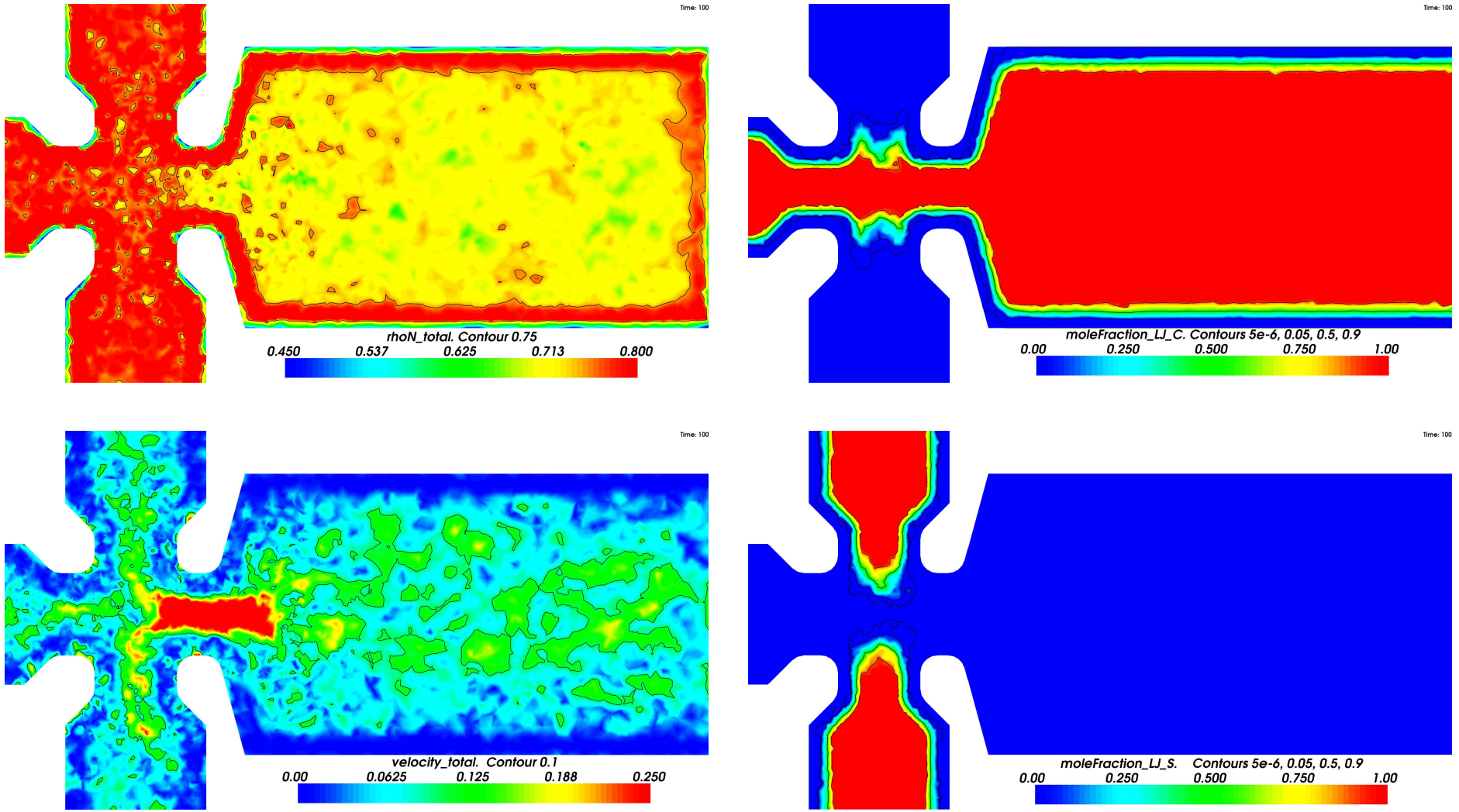


Figure 13.33: Equal potentials, $T = 1.0$, time $= 100$

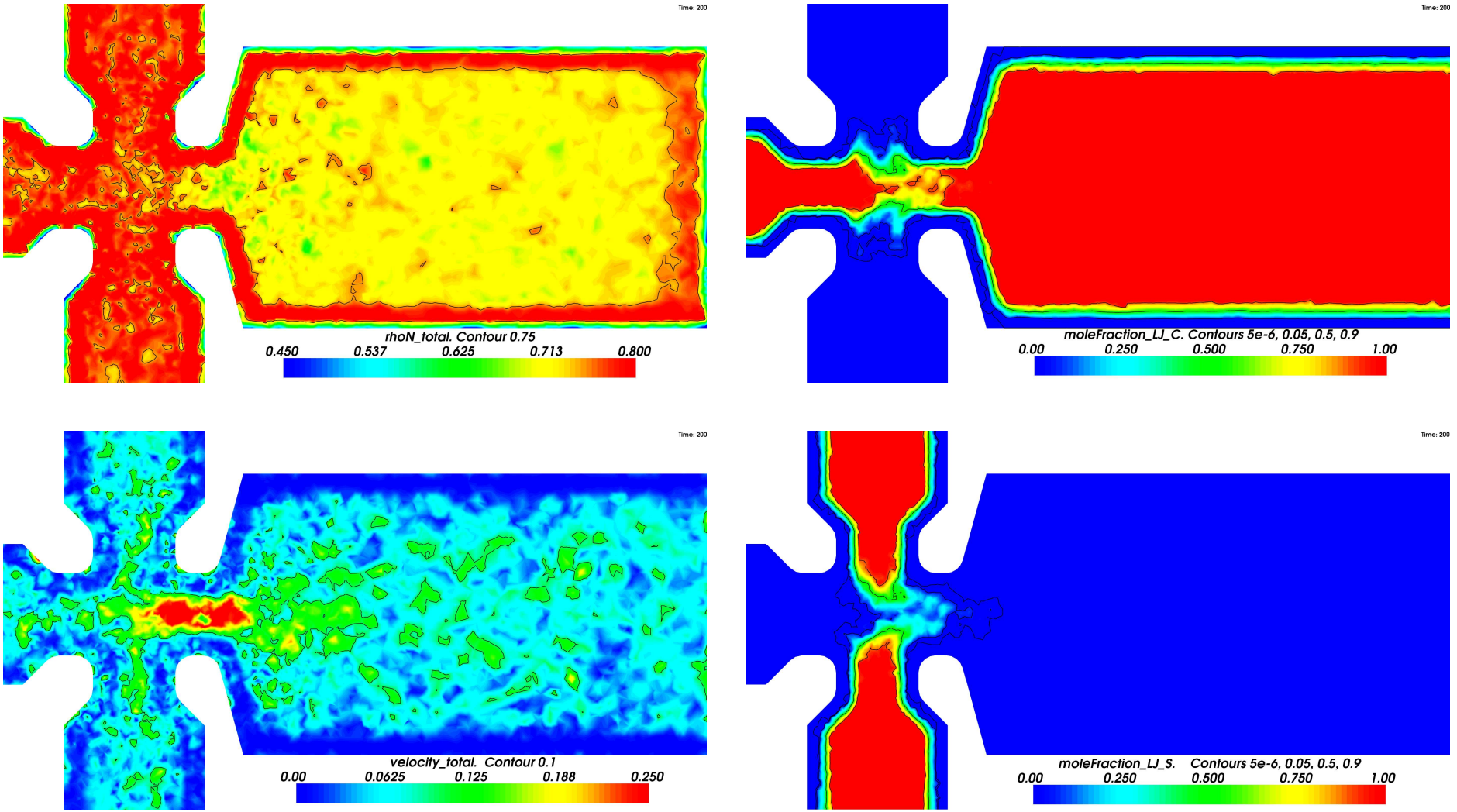


Figure 13.34: Equal potentials, $T = 1.0$, time $= 200$

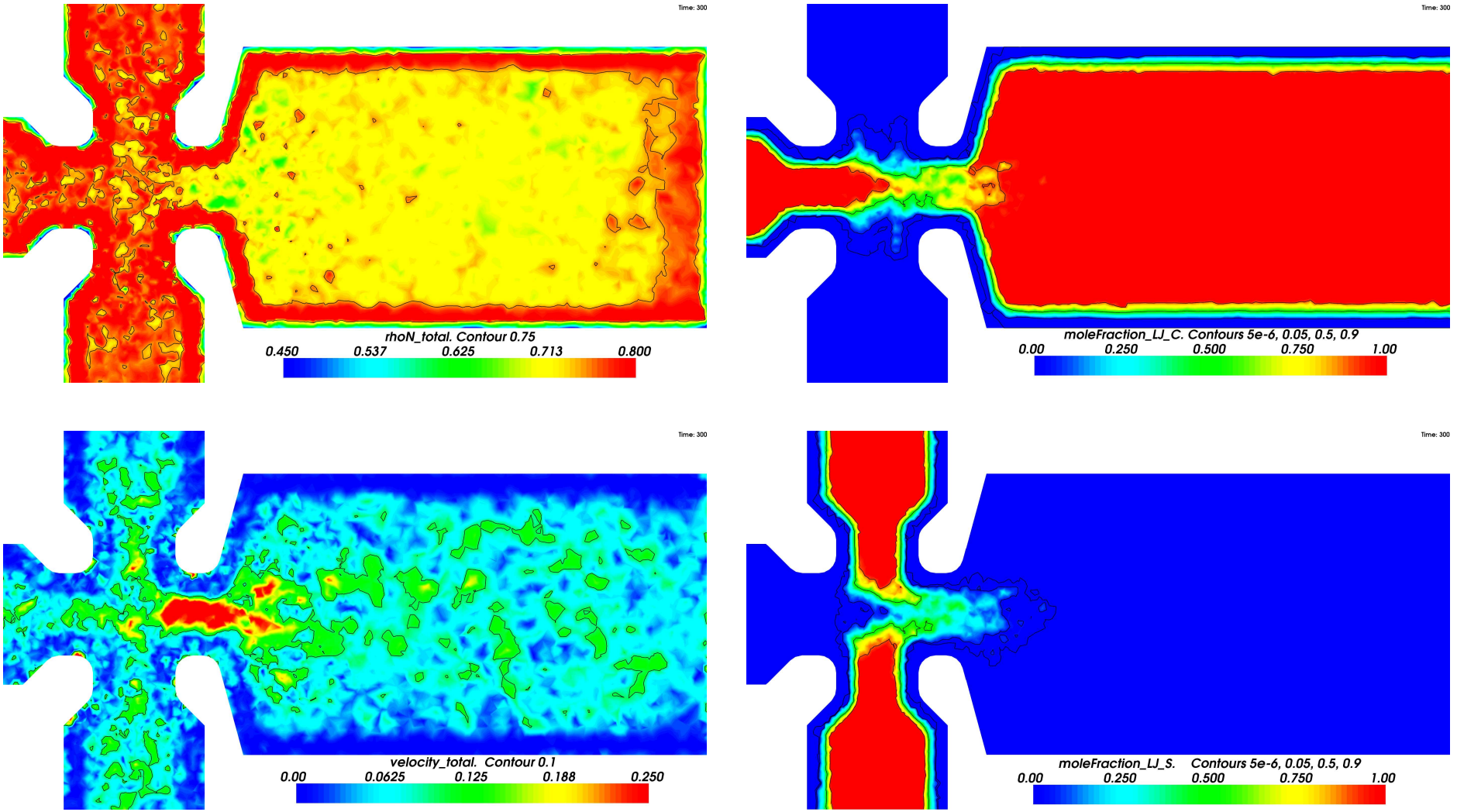


Figure 13.35: Equal potentials, $T = 1.0$, time $= 300$

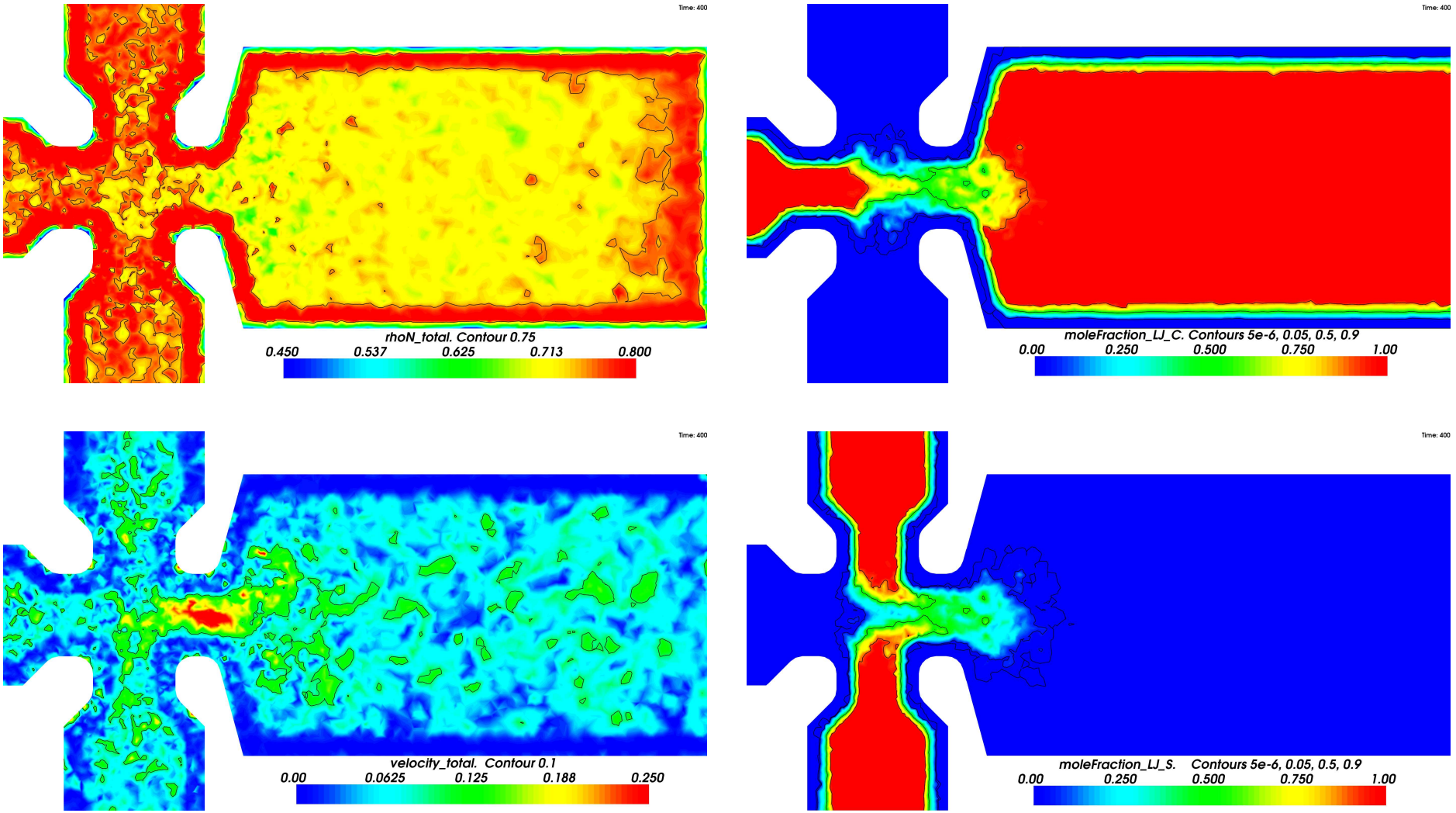


Figure 13.36: Equal potentials, $T = 1.0$, time $= 400$

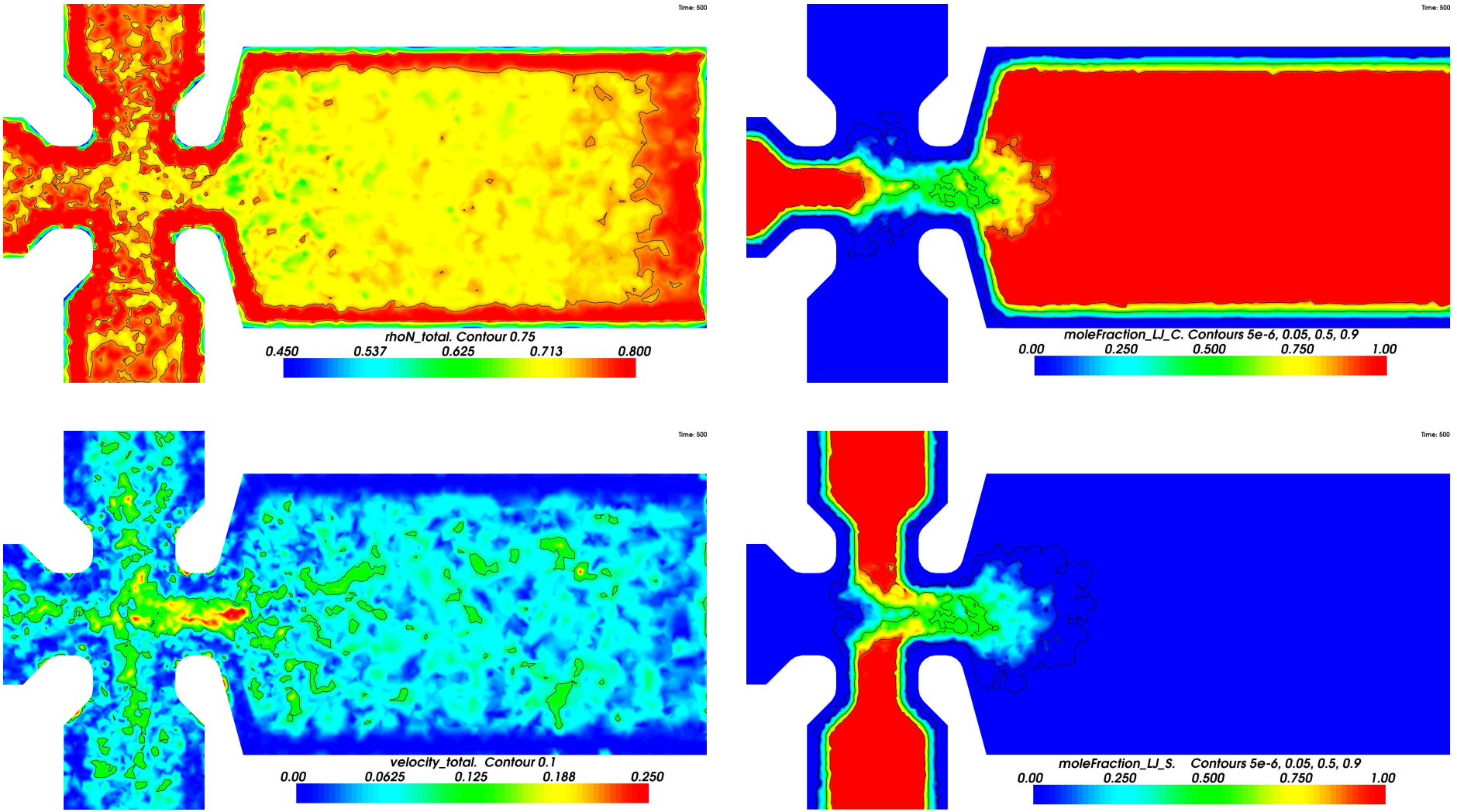


Figure 13.37: Equal potentials, $T = 1.0$, time $= 500$

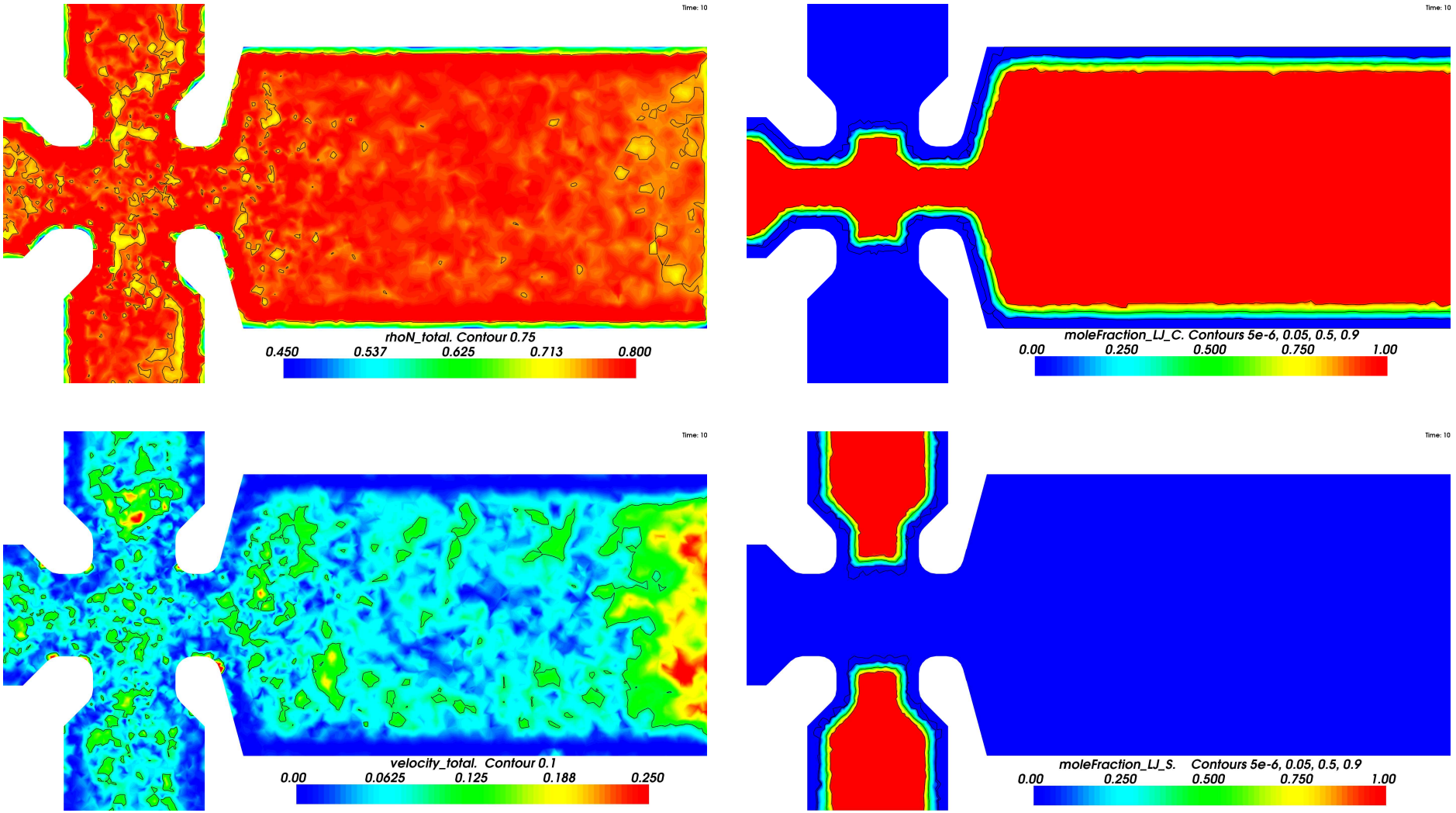


Figure 13.38: Different potentials, $T = 1.0$, time $= 10$

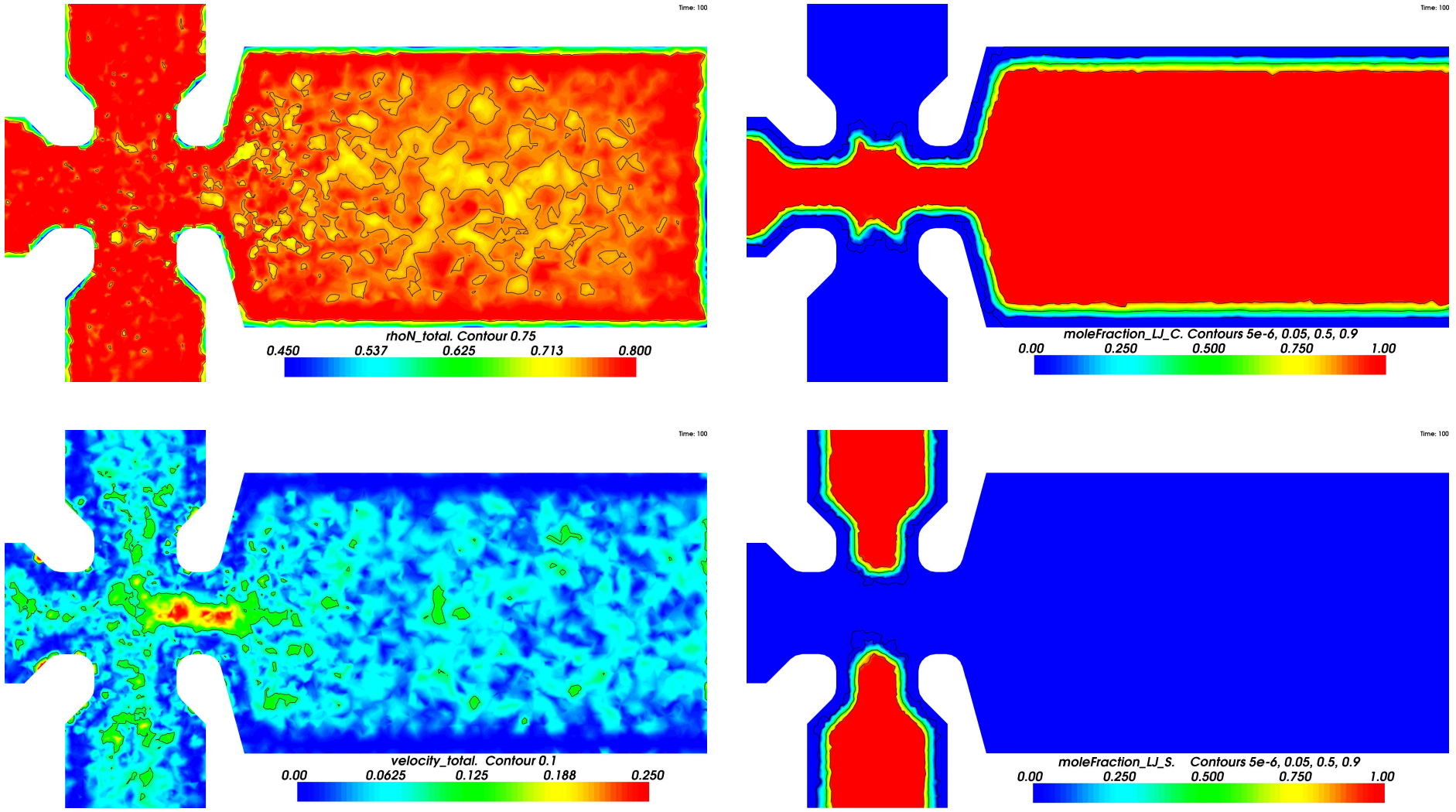


Figure 13.39: Different potentials, $T = 1.0$, time $= 100$

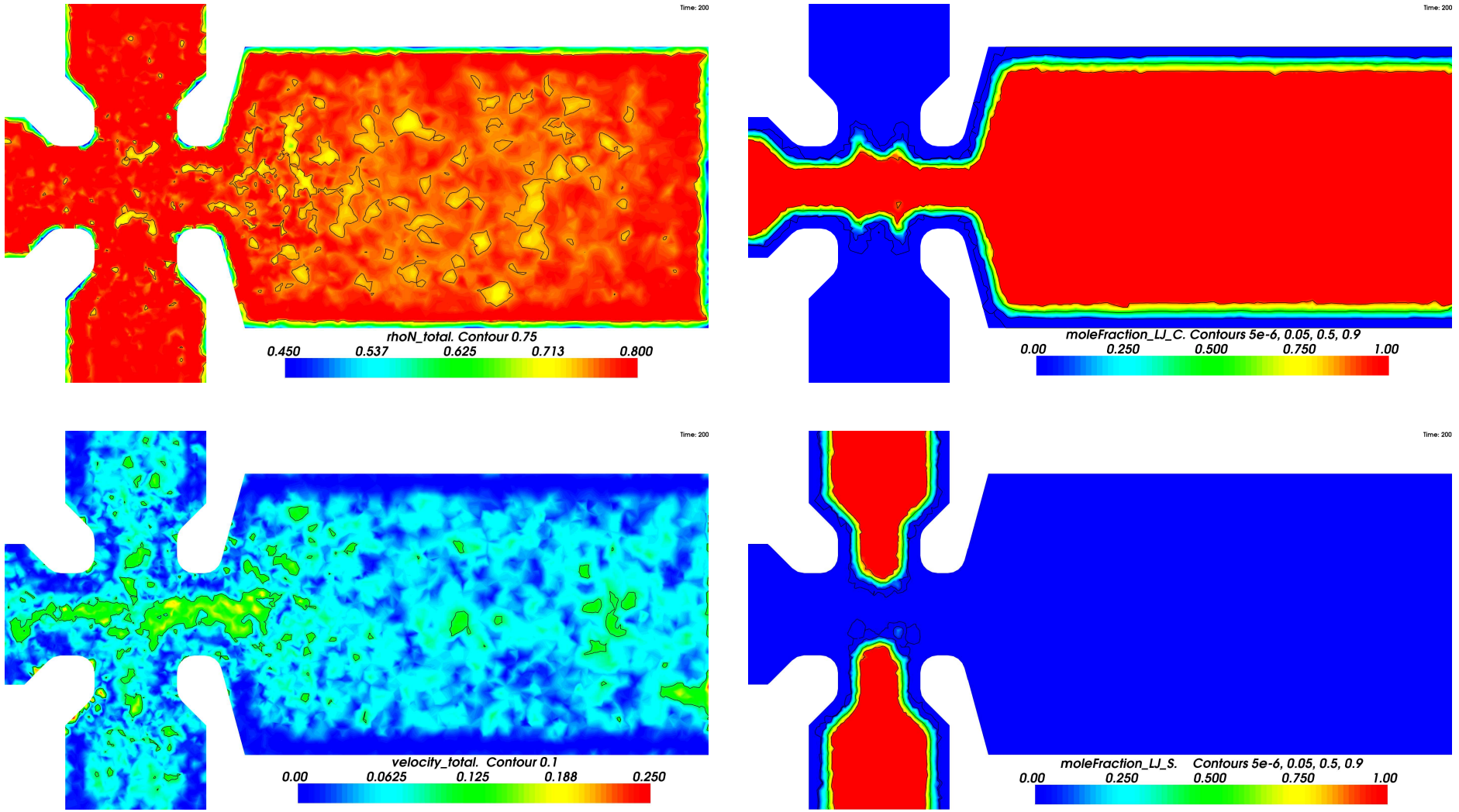


Figure 13.40: Different potentials, $T = 1.0$, time $= 200$

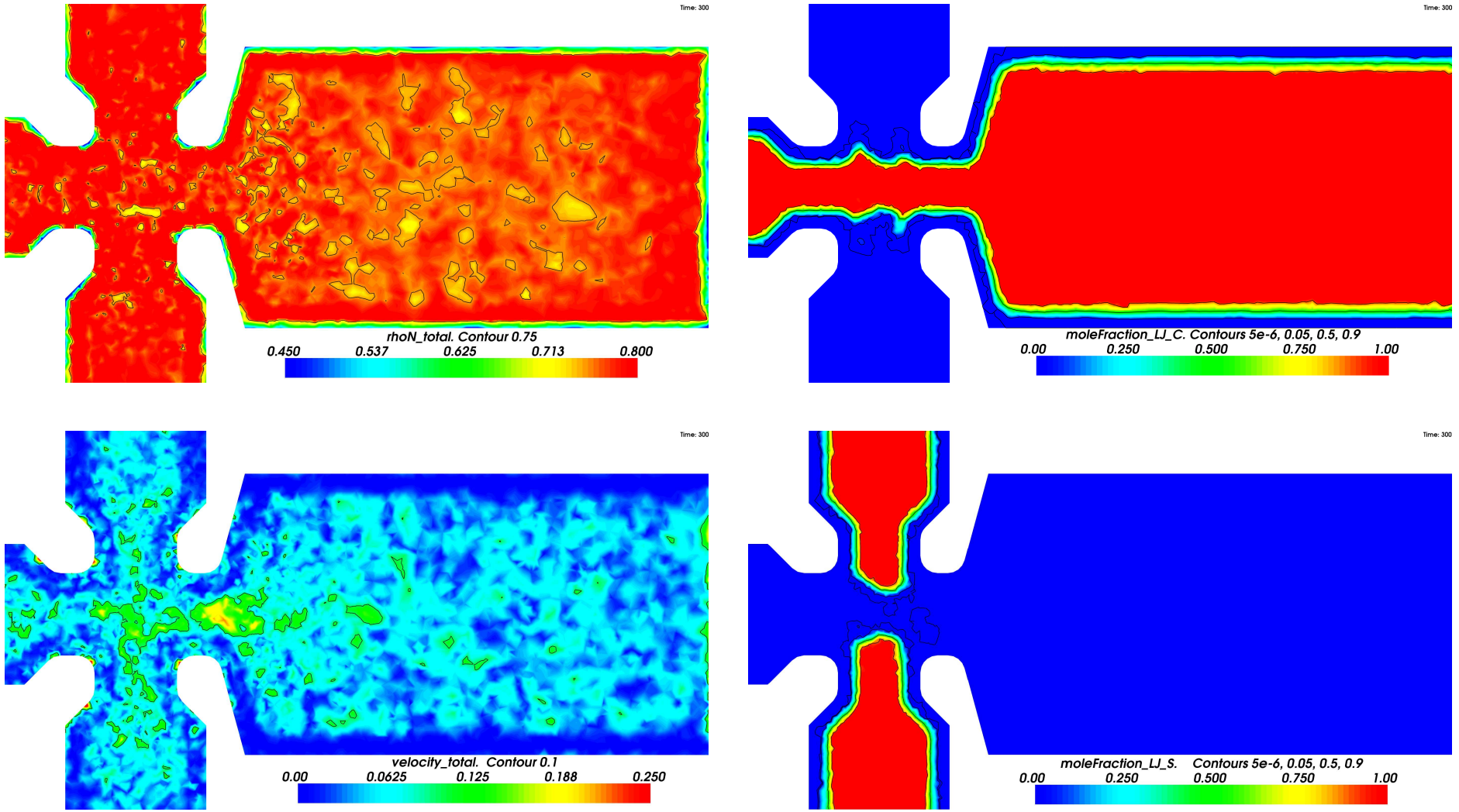


Figure 13.41: Different potentials, $T = 1.0$, time $= 300$

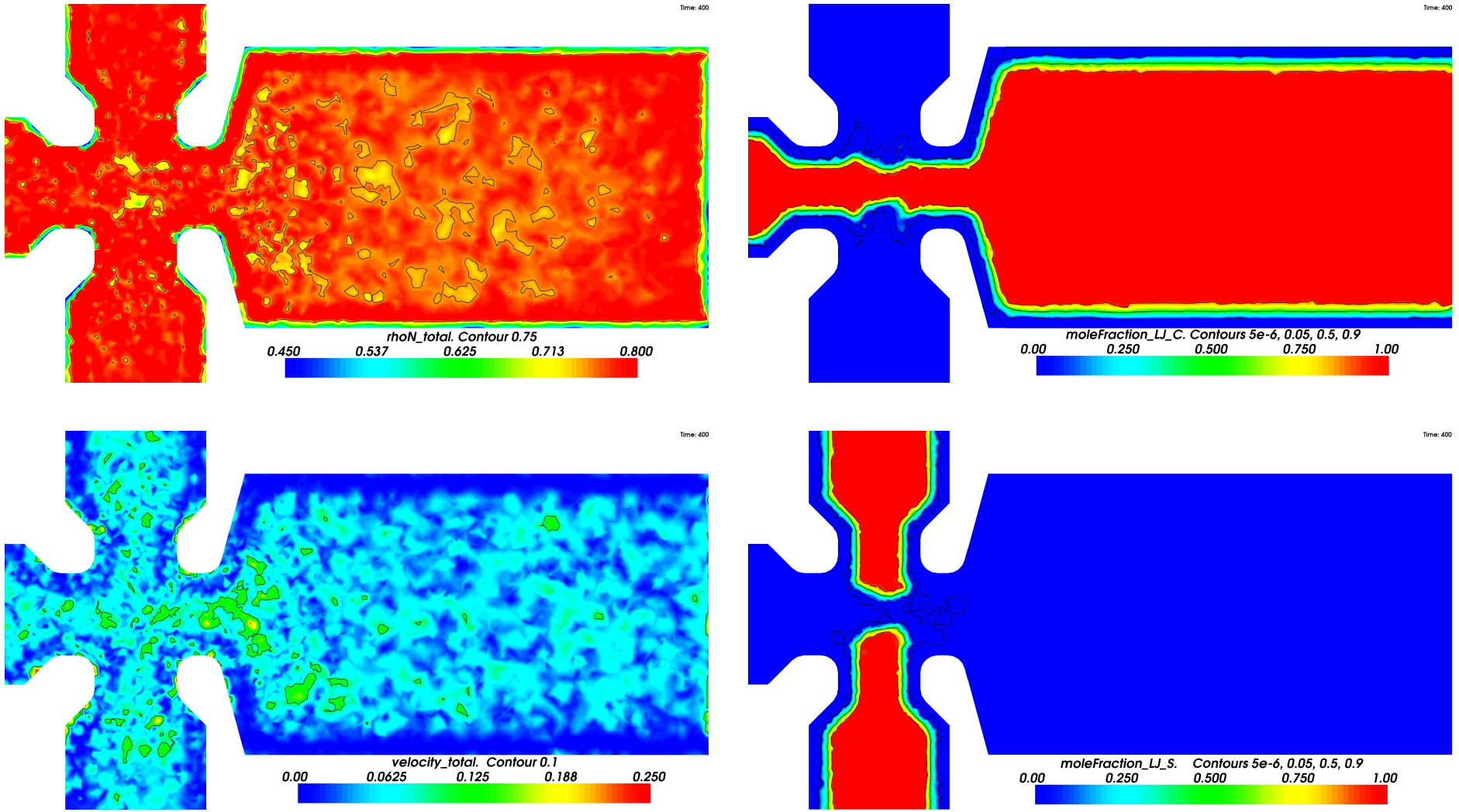


Figure 13.42: Different potentials, $T = 1.0$, time $= 400$

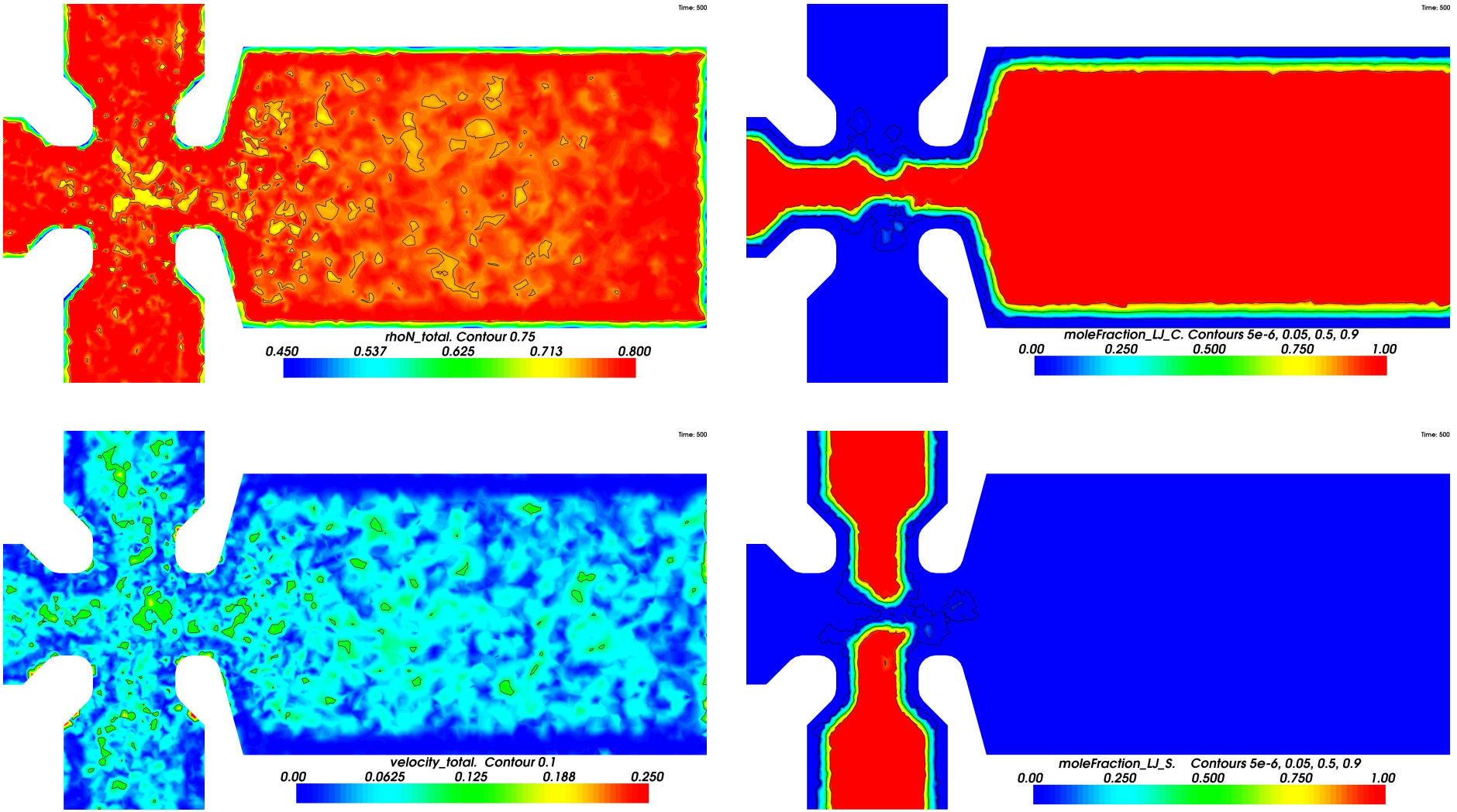


Figure 13.43: Different potentials, $T = 1.0$, time $= 500$

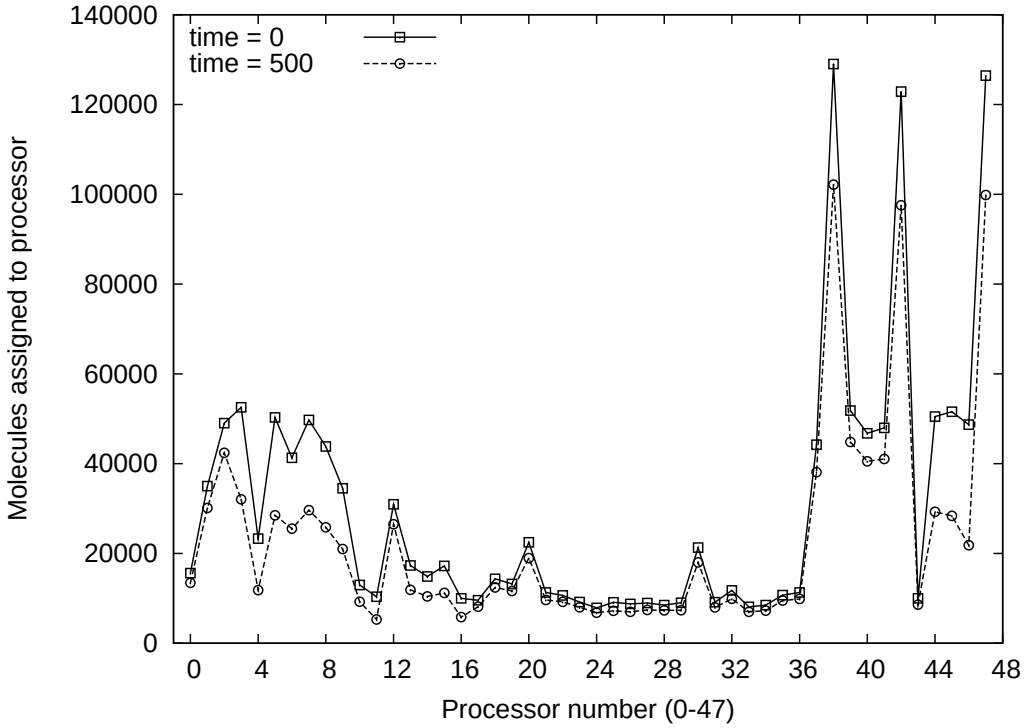


Figure 13.44: The distribution of molecules amongst 48 processors for the $T = 2.5$ case where the potentials are all equal. Some processors simulate significantly more molecules than others. The situation has not changed significantly by the end of the simulation.

13.4 Discussion

A significant performance penalty arises from an unbalanced decomposition of the system. The decomposition was carried out so that the number of cells on each processor was approximately the same, which is desirable for CFD. However, the cells are not the same size, particularly because they have been made deliberately fine in the mixing section. The number of molecules in a portion of the mesh is proportional to the total volume of cells in that mesh portion in this case because density is reasonably homogeneous. The distribution of the number of molecules on each processor is shown in figure 13.44 at the start and end of the $T = 2.5$ equal potential simulation and shows that some processors are simulating significantly more molecules than others. A modification to OpenFOAM's decomposition utility (decomposePar) is required to balance the decomposition.

It is very difficult to make reliable comparisons between the simulations because the fluid state is uncontrolled and the alteration of potentials alters the properties and transport coefficients of the fluid. The use of an outlet venting to

vacuum is also an uncontrolled and unsuitable way to drive the flow; it is likely to be the cause of the density barrier observed in the $T = 1.0$, equal potential simulation. The intention of this case study was to illustrate the potential of the code to perform simulations in complex geometries. It has achieved this aim, and highlighted the need for the infrastructure to be created to make such simulations more controllable.

Chapter 14

Conclusions and further work

14.1 Conclusions

The objective of this research has been to create a molecular dynamics code with functionality not present in those that currently exist: the ability to perform simulations in complex geometries, represented by unstructured meshes of arbitrary polyhedra, such as those derived from engineering CAD models. These meshes are subdivided for parallel processing, and the simulation is insensitive to the size and shape of the decomposed portions, and their connectivity.

The code is intended to serve as a component in a hybrid molecular dynamics-continuum fluid mechanics simulation method, which is the subject of ongoing research, although it may be equally well employed as a standalone MD code for the direct simulation of complex nanoscale structures. More generally, the methods developed may be used for performing any particulate simulation in an arbitrary geometry where long range forces between particles are important, i.e. larger particles that carry an electrostatic charge, such as colloids.

A tool has been written that is able to create initial configurations of molecules by filling spatial zones of the mesh with single crystal lattices of user specified molecule type, structure, density and orientation. This tool operates in parallel, where the mesh has already been decomposed. Each processor fills the zones on its own portion of the mesh, but does so in a way that there are no defects across processor boundaries. Parallelisation of this tool allows large systems of molecules to be created because there is no constraint that all of the molecules must fit into the memory of a single computer at any point.

A molecular dynamics code has been written which calculates pair forces

between molecules which occupy an arbitrary mesh. The computational cost of the force calculation has been kept low, scaling linearly with the number of molecules in the system. This is achieved by determining the spatial relationship of cells once at the start of the simulation and using this information at each timestep to specify which molecules must interact. Replicated molecules are used to pass intermolecular forces across periodic and interprocessor boundaries. Determining how to efficiently implement this force calculation, and particularly how to do it in parallel, comprises the primary original contribution of this work. The motion of molecules is tracked through the mesh, where they interact with boundaries (i.e. inlets, outlets, solid surfaces or interprocessor surfaces) when they cross the appropriate mesh face, making the boundary definition locally specified and able to comprise any shape.

All code has been implemented using OpenFOAM [5]. The most important technical advantages this confers are the powerful and flexible mesh structure and existing particle tracking infrastructure; both of these features (like all of OpenFOAM) were already parallelised. OpenFOAM is written in C++, which allows new functionality to be easily incorporated by deriving new, specialised classes from templated generic classes that contain core functionality. New features are therefore able to be easily made interoperable with the rest of the code infrastructure. OpenFOAM is open source, and as such its use also facilitates the dissemination, and wider use, of the newly developed MD code.

The performance of the initial configuration generation and force calculation algorithms has been shown to scale favourably with the number of cells and molecules in a mesh. A case study of a parallelised, complex geometry containing a large number of molecules, producing complex flow patterns, demonstrates the ability of the code to tackle useful and realistic simulations.

14.2 Future developments

The following is a discussion of tasks required to expand the scope of the MD code to make it applicable to realistic problems. The subject of implementing a hybrid simulation will not be covered in detail.

14.2.1 Expand range of materials

The infrastructure for simulating a wider range of materials needs to be established. This involves generalising the pair potential infrastructure using some of the functionality of the C++ programming language, namely templating, inheritance and virtual functions. The core of the force calculation code can call for the force and energy between a pair of molecules without needing to know which potential is involved or how it is implemented. This allows easy run-time selection of different forms of potential, (i.e. Lennard-Jones, Buckingham etc. [133, 178]) and the ability to add new potential models to the code without changing the force calculation classes. In this way, users who are not acquainted with the core details of the code may make meaningful extensions to it. This is the case with the structure of the rest of OpenFOAM. The potential infrastructure can be further generalised to simulate many body interactions.

Currently the code deals only with short-ranged pair forces between single, spherically symmetrical atoms. Rigid (i.e. diatomic gases, water molecules) and flexible (i.e. hydrocarbons, polymers, biomolecules) rotationally asymmetric polyatomic molecules should be included. This will require predictor-corrector equation of motion integration, incorporating rotational dynamics [130, 131].

The inclusion of long-range electrostatic forces are necessary for many applications. The methods normally used to calculate long range interactions (the Ewald sum [130, 131], smoothed particle-mesh Ewald [179] are applicable to periodic systems only. In complex geometries, the reaction field method [131] or a variant of the fast multipole method [130] with a generalised hierarchical cell subdivision could be investigated.

The range of materials that can be generated by molConfig should be expanded. The fourteen Bravais lattices [180] can be implemented to allow real, non-homogeneous crystals to be filled into volumes. A library of common solids and liquids drawn from the open literature should be established. The stochastic creation of mixtures would be useful, for example including a specified proportion of a dissolved gas, randomly distributed in a liquid.

14.2.2 Measurements

The range of properties that are measured should be expanded, particularly to include stress and heat flux. A flexible infrastructure is required to make spe-

cialised measurements, in addition to simply resolving properties onto the mesh. A surface may be defined and the profile of a flux passing through it measured with high spatial resolution. Similarly, a volume may be defined and the variation of a parameter in a specified direction measured with high spatial resolution. Simpler measurements (the total flux of a certain species through a plane for example) can be made with the same methods. As with generalised potential functions, this infrastructure should be designed so that new measurements may be made without needing to alter the core force calculation and molecule motion code.

Block averaging methods [130] should be applied to time averaged quantities to allow an accurate value for the variance in a measurement to be estimated. Overlapping data collection [130] can be used to reduce the noise in time varying data without increasing the result output interval.

An automated tool is required to measure fluid transport properties over a specified range of states to ensure that continuum simulations that the MD simulation is hybridised with, or compared to, represent the same fluid as the MD potential.

14.2.3 Boundaries and driven flows

Methods are required to create controlled inlet and outlet boundaries, as well as creating regions of the domain where the state of the fluid is controlled. This is required to apply the type of flow boundary conditions that are common in CFD: controlling the velocity, pressure and temperature of the fluid. Volumes where a fluid state is controlled can use the same underlying infrastructure as those used to make measurements. This functionality is also required for coupling a hybrid simulation.

14.2.4 Performance improvements

The parallel performance of the code can be improved by profiling to determine the most costly functions and optimising them. The decomposition of the mesh should be altered to weight a cell by the number of molecules it contains in order to equally distribute the number of molecules among processors. A method of repartitioning cells and molecules between processors to dynamically load-balance the simulation during running would be beneficial in transient simulations where

large density variations move through the mesh.

The performance impact of parallel communications latency can be greatly reduced by communicating referred molecules between processors during the force calculation between pairs of real molecules.

Alternative methods for searching the mesh to establish which cells are in range of each other can be tried to attempt to reduce the time taken to build interaction lists. For example, a 'wave' can be launched into the mesh from the faces of the cell in question which moves outwards and stops propagating when the last cell it moved through is not in range. This would avoid needing to search the whole mesh.

Bibliography

[1] G. B. Macpherson, M. K. Borg and J. M. Reese. Parallel generation of molecular dynamics initial configurations in arbitrary geometries. *Molecular Simulation*, 33(15): 1199–1212, 2007.

[2] G. B. Macpherson and J. M. Reese. Molecular dynamics in arbitrary geometries: parallel evaluation of pair forces. *Molecular Simulation*, 34(1): 97–115, 2008.

[3] G. B. Macpherson, N. Nordin and H. G. Weller. Particle tracking in unstructured, arbitrary polyhedral meshes for use in CFD and molecular dynamics. *Communications in Numerical Methods in Engineering*, DOI: 10.1002/cnm.1128, 2008.

[4] G. B. Macpherson and J. M. Reese. Molecular dynamics for near surface flows in nano liquid and microgas systems. *Proceedings of the 4th ASME International Conference on Nanochannels, Microchannels and Minichannels*. 2006. Limerick, Eire.

[5] OpenFOAM: The Open Source CFD Toolbox. http://www.openfoam.org.

[6] R. P. Feynman. There's plenty of room at the bottom, 1959. http://www.its.caltech.edu/∼feynman/plenty.html.

[7] P. J. Kemery, J. K. Steehler and P. W. Bohn. Electric field mediated transport in nanometer diameter channels. *Langmuir*, 14(10): 2884–2889, 1998.

[8] P. Agre. The aquaporin water channels. *Proceedings of the American Thoracic Society*, 3(1): 5–13, 2006.

[9] J. Koplik and J. R. Banavar. Continuum deductions from molecular hydrodynamics. *Annual Review of Fluid Mechanics*, 27: 257–292, 1995.

[10] P. A. Thompson and M. O. Robbins. Shear flow near solids: Epitaxial order and flow boundary conditions. *Physical Review A*, 41: 6830–6837, 1990.

[11] S. Duhr and D. Braun. Why molecules move along a temperature gradient. *Proceedings of the National Academy of Sciences*, 103(52): 19678–19682, 2006.

[12] R. D. Astumian. Coupled transport at the nanoscale: the unreasonable effectiveness of equilibrium theory. *Proceedings of the National Academy of Sciences*, 104(1): 3–4, 2007.

[13] K. Travis, B. Todd and D. Evans. Poiseuille flow of molecular fluids. *Physica A*, 240(1-2): 315–27, 1997.

[14] K. Travis and K. Gubbins. Poiseuille flow of Lennard-Jones fluids in narrow slit pores. *Journal of Chemical Physics*, 112(4): 1984–94, 2000.

[15] B. J. Kirby and E. F. Hasselbrink. Zeta potential of microfluidic substrates: 1. theory, experimental techniques, and effects on separations. *Electrophoresis*, 25: 187–202, 2004.

[16] J. N. Israelachvili. *Intermolecular and Surface Forces: with Applications to Colloidal and Biological Systems*. Academic Press, 2nd edition, 1991.

[17] A. T. Conlisk, J. McFerran, Z. Zheng and D. Hansford. Mass transfer and flow in electrically charged micro- and nanochannels. *Analytical Chemistry*, 74(9): 2139–2150, 2002.

[18] R. Sadr, M. Yoda, Z. Zheng and A. T. Conlisk. An experimental study of electro-osmotic flow in rectangular microchannels. *Journal of Fluid Mechanics*, 506: 357–367, 2004.

[19] R. Karnik, K. Castelino and A. Majumdar. Field-effect control of protein transport in a nanofluidic transistor circuit. *Applied Physics Letters*, 88(12): 123114, 2006.

[20] I. Vlassiouk and Z. Siwy. Nanofluidic diode. *Nano Letters*, 7(3): 552–556, 2007.

[21] R. Karnik, R. Fan, M. Yue, D. Li, P. Yang and A. Majumdar. Electrostatic control of ions and molecules in nanofluidic transistors. *Nano Letters*, 5(5): 943–948, 2005.

[22] K. Fa, J. Tulock, J. Sweedler and P. Bohn. Profiling pH gradients across nanocapillary array membranes connecting microfluidic channels. *Journal of the American Chemical Society*, 127(40): 13928–13933, 2005.

[23] S. Pennathur and J. G. Santiago. Electrokinetic transport in nanochannels. 2. Experiments. *Analytical Chemistry*, 77(21): 6782–6789, 2005.

[24] G. Karniadakis, A. Beskok and N. Aluru. *Microflows and Nanoflows: Fundamentals and Simulation*. Springer, 2005.

[25] M. Whitby and N. Quirke. Fluid flow in carbon nanotubes and nanopipes. *Nature Nanotechnology*, 2: 87–94, 2007.

[26] M. Majumder, N. Chopra, R. Andrews and B. J. Hinds. Nanoscale hydrodynamics: Enhanced flow in carbon nanotubes. *Nature*, 438: 44, 2005.

[27] S. Supple and N. Quirke. Rapid imbibition of fluids in carbon nanotubes. *Physical Review Letters*, 90(21): 214501, 2003.

[28] S. Supple and N. Quirke. Molecular dynamics of transient oil flows in nanopores I: imbibition speeds for single wall carbon nanotubes. *Journal of Chemical Physics*, 121(17): 8571–8579, 2004.

[29] S. Supple and N. Quirke. Molecular dynamics of transient oil flows in nanopores. II. density profiles and molecular structure for decane in carbon nanotubes. *Journal of Chemical Physics*, 122(10): 104706, 2005.

[30] M. Longhurst and N. Quirke. The environmental effect on the radial breathing mode of carbon nanotubes in water. *Journal of Chemical Physics*, 124(23): 234708, 2006.

[31] M. Longhurst and N. Quirke. The environmental effect on the radial breathing mode of carbon nanotubes. II. Shell model approximation for internally and externally adsorbed fluids. *Journal of Chemical Physics*, 125(18): 184705, 2006.

[32] E. W. Washburn. The dynamics of capillary flow. *Physical Review*, 17(3): 273–283, 1921.

[33] M. Majumder, X. Zhan, R. Andrews and B. J. Hinds. Voltage gated carbon nanotube membranes. *Langmuir*, 23(16): 8624–8631, 2007.

[34] M. P. Rossi, H. Ye, Y. Gogotsi, S. Babu, P. Ndungu and J. C. Bradley. Environmental scanning electron microscopy study of water in carbon nanopipes. *Nano Letters*, 4(5): 989–993, 2004.

[35] Y. Qiao, G. Cao and X. Chen. Effects of gas molecules on nanofluidic behaviors. *Journal of the American Chemical Society*, 129(8): 2355–2359, 2007.

[36] H. H. Bau, S. Sinha, B. Kim and M. Riegelman. Fabrication of nanofluidic devices and the study of fluid transport through them. *Proceedings of SPIE. Nanofabrication: Technologies, Devices, and Applications*, volume 5592, pages 201–213. SPIE, 2005.

[37] M. Maccarini. Water at solid surfaces: A review of selected theoretical aspects and experiments on the subject. *Biointerphases*, 2(3): MR1–MR15, 2007.

[38] J. Israelachvili and H. Christenson. Liquid structure and short-range forces between surfaces in liquids. *Physica A*, 140A: 278–284, 1986.

[39] O. Vinogradova. Drainage of a thin liquid film confined between hydrophobic surfaces. *Langmuir*, 11(6): 2213–2220, 1995.

[40] T. Becker and F. Mugele. Nanofluidics: viscous dissipation in layered liquid films. *Physical Review Letters*, 91(16): 166104, 2003.

[41] H. K. Christenson and P. M. Claesson. Direct measurements of the force between hydrophobic surfaces in water. *Advances in Colloid and Interface Science*, 91(3): 391–436, 2001.

[42] M. Gad–el–Hak. The fluid mechanics of microdevices — The Freeman Scholar lecture. *Journal of Fluids Engineering*, 121: 5–33, 1999.

[43] J. C. Maxwell. On stresses in rarified gases arising from inequalities of temperature. *Philosophical Transactions of the Royal Society of London, Part 1*, 170: 231–256, 1879.

[44] E. Lauga, M. P. Brenner and H. A. Stone. Microfluidics: The no-slip boundary condition. Chapter 19 in Handbook of Experimental Fluid Dynamics, or http://arxiv.org/abs/cond-mat/0501557, 2005.

[45] S. Granick, Y. Zhu and H. Lee. Slippery questions about complex fluids flowing past solids. *Nature Materials*, 2(4): 221–227, 2003.

[46] Y. Zhu and S. Granick. Rate-dependent slip of newtonian liquid at smooth surfaces. *Physical Review Letters*, 87(9): 96105, 2001.

[47] Y. Zhu and S. Granick. Limits of the hydrodynamic no-slip boundary condition. *Physical Review Letters*, 88: 106102, 2002.

[48] H. Spikes and S. Granick. Equation for slip of simple liquids at smooth solid surfaces. *Langmuir*, 19(12): 5065–5071, 2003.

[49] E. Lauga and M. P. Brenner. Dynamic mechanisms for apparent slip on hydrophobic surfaces. *Physical Review E*, 70: 026311, 2004.

[50] D. C. Tretheway and C. D. Meinhart. A generating mechanism for apparent fluid slip in hydrophobic microchannels. *Physics of Fluids*, 16(5): 1509–1515, 2004.

[51] D. C. Tretheway, X. Liu and C. D. Meinhart. Analysis of slip flow in microchannels. Technical report, Department of Mechanical Engineering, University of California Santa Barbara, 2002.

[52] C. Cottin-Bizonne, B. Cross, A. Steinberger and E. Charlaix. Boundary slip on smooth hydrophobic surfaces: Intrinsic effects and possible artifacts. *Physical Review Letters*, 94: 056102, 2005.

[53] V. S. J. Craig, C. Neto and D. R. M. Williams. Shear-dependent boundary slip in an aqueous newtonian liquid. *Physical Review Letters*, 87(5): 054504, 2001.

[54] C.-H. Choi, K. J. A. Westin and K. S. Breuer. Apparent slip flows in hydrophilic and hydrophobic microchannels. *Physics of Fluids*, 15(10): 2897–2902, 2003.

[55] P. Thompson and S. Troian. A general boundary condition for liquid flow at solid surfaces. *Nature*, 389(6649): 360–362, 1997.

[56] S. Lichter, A. Martini, R. Q. Snurr and Q. Wang. Liquid slip in nanoscale channels as a rate process. *Physical Review Letters*, 98(22): 226001, 2007.

[57] M. Cieplak, J. Koplik and J. R. Banavar. Boundary conditions at a fluid-solid interface. *Physical Review Letters*, 86(5): 803–806, 2001.

[58] J. W. G. Tyrrell and P. Attard. Images of nanobubbles on hydrophobic surfaces and their interactions. *Physical Review Letters*, 87: 176104, 2001.

[59] J. Tyrrell and P. Attard. Atomic force microscope images of nanobubbles on a hydrophobic surface and corresponding force-separation data. *Langmuir*, 18(1): 160–167, 2002.

[60] P. Attard, M. P. Moody and J. W. G. Tyrrell. Nanobubbles: the big picture. *Physica A*, 314: 696–705, 2002.

[61] X. Zhang, N. Maeda and V. Craig. Physical properties of nanobubbles on hydrophobic surfaces in water and aqueous solutions. *Langmuir*, 22(11): 5025–5035, 2006.

[62] D. A. Doshi, E. B. Watkins, J. N. Israelachvili and J. Majewski. Reduced water density at hydrophobic surfaces: Effect of dissolved gases. *Proceedings of the National Academy of Sciences*, 102(27): 9458–9462, 2005.

[63] A. Poynor, L. Hong, I. K. Robinson, S. Granick, Z. Zhang and P. A. Fenter. How water meets a hydrophobic surface. *Physical Review Letters*, 97(26): 266101, 2006.

[64] Z. Ge, D. G. Cahill and P. V. Braun. Thermal conductance of hydrophilic and hydrophobic interfaces. *Physical Review Letters*, 96(18): 186101, 2006.

[65] Y. Zhu and S. Granick. Limits of the hydrodynamic no-slip boundary condition. *Physical Review Letters*, 88(10): 106102, 2002.

[66] M. O. Jensen, O. G. Mouritsen and G. H. Peters. The hydrophobic effect: Molecular dynamics simulations of water confined between extended hydrophobic and hydrophilic surfaces. *Journal of Chemical Physics*, 120(20): 9729–9744, 2004.

[67] P. Setny and M. Geller. Water properties inside nanoscopic hydrophobic pocket studied by computer simulations. *Journal of Chemical Physics*, 125(14): 144717, 2006.

[68] D. Evans, E. Cohen and G. Morriss. Probability of second law violations in shearing steady states. *Physical Review Letters*, 71(15): 2401–2404, 1993.

[69] G. Aytona, D. J. Evans and D. J. Searles. A local fluctuation theorem. *Journal of Chemical PhysicS*, 115(5): 2033–2037, 2001.

[70] G. M. Wang, E. M. Sevick, E. Mittag, D. J. Searles and D. J. Evans. Experimental demonstration of violations of the second law of thermodynamics for small systems and short time scales. *Physical Review Letters*, 89(5): 050601, 2002.

[71] D. J. Evans and D. J. Searles. The fluctuation theorem. *Advances in Physics*, 51(7): 15291585, 2002.

[72] E. Mittag and D. J. Evans. Time-dependent fluctuation theorem. *Physical Review E*, 67: 026113, 2003.

[73] L. D. Landau and E. M. Lifshitz. *Fluid mechanics (Course of Theoretical Physics v.6)*. Pergamon Press, 2nd edition, 1987.

[74] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 3rd edition, 2002.

[75] J. D. Anderson. *Computational Fluid Dynamics. The Basics with Applications.* McGraw-Hill, 1995.

[76] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics.* Prentice Hall, 2nd edition, 2007.

[77] G. De Fabritiis, M. Serrano, R. Delgado-Buscalioni and P. Coveney. Fluctuating hydrodynamic modeling of fluids at the nanoscale. *Physical Review E*, 75(2): 26307, 2007.

[78] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Springer-Verlag, 1992.

[79] L. D. Landau and E. M. Lifshitz. *Statistical Physics Part 2 (Course of Theoretical Physics v.9).* Pergamon, 1980.

[80] R. Kubo. The fluctuation-dissipation theorem. *Reports on Progress in Physics*, 29: 255–284, 1966.

[81] K. Kadau, T. C. Germann and P. S. Lomdahl. Large scale molecular dynamics simulation of 19 billion particles. *International Journal of Modern Physics C*, 15(1): 193–201, 2004.

[82] K. Kadau, T. C. Germann, N. G. Hadjiconstantinou, P. S. Lomdahl, G. Dimonte, B. L. Holian and B. J. Alder. Nanohydrodynamics simulations: An atomistic view of the Rayleigh-Taylor instability. *Proceedings of the National Academy of Sciences of the United States of America*, 101(16): 5851–5855, 2004.

[83] T. Werder, J. H. Walther and P. Koumoutsakos. Hybrid atomistic-continuum method for the simulation of dense fluid flows. *Journal of Computational Physics*, 205(1): 373–390, 2005.

[84] S. Succi. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond.* Clarendon Press, 2001.

[85] J. Horbach and S. Succi. Lattice boltzmann versus molecular dynamics simulation of nanoscale hydrodynamic flows. *Physical Review Letters*, 96(22): 224503, 2006.

[86] G. A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Oxford University Press, 1994.

[87] M. N. Kogan. *Rarefied Gas Dynamics.* Plenum Press, 1969.

[88] R. Delgado-Buscalioni and P. V. Coveney. Hybrid molecular-continuum fluid dynamics. *Philosophical Transactions of the Royal Society London A*, 362(1821): 1639–1654, 2004.

[89] E. G. Flekkøy, P. V. Coveney and G. De Fabritiis. Foundations of dissipative particle dynamics. *Physical Review E*, 62(2A): 2140–2157, 2000.

[90] P. Español and P.Warren. Statistical mechanics of dissipative particle dynamics. *Europhysics Letters*, 30(4): 191–196, 1995.

[91] P. Español and M. Revenga. Smoothed dissipative particle dynamics. *Physical Review E*, 67(2): 026705, 2003.

[92] P. Español, M. Serrano and H. C. Öttinger. Thermodynamically admissible form for discrete hydrodynamics. *Physical Review Letters*, 83: 4542–4545, 1999.

[93] P. Español. *Trends in Nanoscale Mechanics: Analysis of Nanostructured Materials and Multi-Scale Modeling*, chapter 1: Dissipative Particle Dynamics. Kluwer, 2003.

[94] M. Serrano and P. Español. Thermodynamically consistent mesoscopic fluid particle model. *Physical Review E*, 64: 046115, 2001.

[95] M. Serrano, G. De Fabritiis, P. Español, E. Flekkøy and P. Coveney. Mesoscopic dynamics of voronoi fluid particles. *Journal of Physics A*, 35(7): 1605–1625, 2002.

[96] G. De Fabritiis and P. Coveney. Dynamical geometry for multiscale dissipative particle dynamics. *Computer Physics Communications*, 153(2): 209–226, 2003.

[97] G. De Fabritiis, P. Coveney and E. Flekkøy. Multiscale dissipative particle dynamics. *Philosophical Transactions of the Royal Society London A*, 360(1792): 317–313, 2002.

[98] G. de Fabritiis, M. Serrano, P. Español and P. V. Coveney. Efficient numerical integrators for stochastic models. *Physica A*, 361: 429–440, 2006.

[99] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[100] H. C. Öttinger. *Beyond Equilibrium Thermodynamics*. Wiley, 2005.

[101] H. C. Öttinger. Nonequilibrium thermodynamics - a tool for applied rheologists. *Applied Rheology*, 9: 17–26, 1999.

[102] M. Grmela and H. C. Öttinger. Dynamics and thermodynamics of complex fluids. I. Development of a general formalism. *Physical Review E*, 56(6): 6620–6632, 1997.

[103] H. C. Öttinger and M. Grmela. Dynamics and thermodynamics of complex fluids. II. Illustrations of a general formalism. *Physical Review E*, 56(6): 6633–6655, 1997.

[104] H. C. Öttinger. General projection operator formalism for the dynamics and thermodynamics of complex fluids. *Physical Review E*, 57(2): 1416–1420, 1998.

[105] J. J. de Pablo and H. C. Öttinger. An atomistic approach to general equation for the nonequilibrium reversible-irreversible coupling. *Journal of Non-Newtonian Fluid Mechanics*, 96(1-2): 137–162, 2001.

[106] D. Hirshfeld and D. Rapaport. Molecular dynamics simulation of Taylor-Couette vortex formation. *Physical Review Letters*, 80(24): 5337–5340, 1998.

[107] D. Rapaport and E. Clementi. Eddy formation in obstructed fluid flow: a molecular-dynamics study. *Physical Review Letters*, 57(6): 695–698, 1986.

[108] D. Rapaport. Microscale hydrodynamics: discrete-particle simulation of evolving flow patterns. *Physical Review A*, 36(7): 3288–3299, 1987.

[109] D. Rapaport. Hexagonal convection patterns in atomistically simulated fluids. *Physical Review E*, 73(2): 25301, 2006.

[110] K. Travis, B. Todd and D. Evans. Departure from Navier-Stokes hydrodynamics in confined liquids. *Physical Review E*, 55(4): 4288–95, 1997.

[111] H. Okumura and D. M. Heyes. Comparisons between molecular dynamics and hydrodynamics treatment of nonstationary thermal processes in a liquid. *Physical Review E*, 70(6): 061206, 2004.

[112] H. Okumura and D. M. Heyes. Stationary temperature profiles in a liquid nanochannel: Comparisons between molecular-dynamics simulation and classical hydrostatics. *Physical Review E*, 74(6): 061201, 2006.

[113] T. Qian and X.-P. Wang. Driven cavity flow: from molecular dynamics to continuum hydrodynamics. *Multiscale Modeling and Simulation*, 3(4): 749–763, 2005.

[114] S. T. O'Connell and P. A. Thompson. Molecular dynamics–continuum hybrid computations: A tool for studying complex fluid flows. *Physical Review E*, 52: R5792–R5795, 1995.

[115] G. Wagner and E. G. Flekkøy. Hybrid computations with flux exchange. *Philosophical Transactions of the Royal Society London A*, 362(1821): 1655–1665, 2004.

[116] X. B. Nie, S. Y. Chen, W. E and M. O. Robbins. A continuum and molecular dynamics hybrid method for micro- and nano-fluid flow. *Journal of Fluid Mechanics*, 500: 55–64, 2004.

[117] G. De Fabritiis, R. Delgado-Buscalioni and P. Coveney. Multiscale modeling of liquids with molecular specificity. *Physical Review Letters*, 97(13): 134501, 2006.

[118] F. Zhu, E. Tajkhorshid and K. Schulten. Theory and simulation of water permeation in aquaporin-1. *Biophysical Journal*, 86(1): 50–57, 2004.

[119] W. Ren and W. E. Heterogeneous multiscale method for the modeling of complex fluids and micro-fluidics. *Journal of Computational Physics*, 204(1): 1–26, 2005.

[120] W. E, B. Engquist and Z. Huang. Heterogeneous multiscale method: a general methodology for multiscale modeling. *Physical Review B*, 67(9): 92101 – 1, 2003.

[121] J. Hedengren and T. Edgar. In situ adaptive tabulation for real-time control. *Industrial & Engineering Chemistry Research*, 44(8): 2716–2724, 2005.

[122] J. N. Israelachvili. Adhesion forces between surfaces in liquids and condensable vapours. *Surface Science Report*, 14(3): 109–159, 1992.

[123] X.-J. Fan, N. Phan-Thien, N. T. Yong and X. Diao. Molecular dynamics simulation of a liquid in a complex nano channel flow. *Physics of Fluids*, 14(3): 1146–1153, 2002.

[124] J. P. Hansen and I. R. McDonald. *Theory of Simple Liquids*. Academic Press, 3rd edition, 2006.

[125] J. M. Kincaid and E. G. D. Cohen. Deviations from hydrodynamics on the nanoscale. *Molecular Physics*, 100(8): 3005–3010, 2002.

[126] H. S. Wijesinghe and N. G. Hadjiconstantinou. Discussion of hybrid atomistic-continuum methods for multiscale hydrodynamics. *International Journal of Multiscale Computational Engineering*, 2: 189–202, 2004.

[127] R. Delgado-Buscalioni and P. V. Coveney. Continuum-particle hybrid coupling for mass, momentum, and energy transfers in unsteady fluid flow. *Physical Review E*, 67(4): 046704, 2003.

[128] R. Delgado-Buscalioni and P. V. Coveney. USHER: an algorithm for particle insertion in dense fluids. *Journal of Chemical Physics*, 119(2): 978–987, 2003.

[129] G. De Fabritiis, R. Delgado-Buscalioni and P. V. Coveney. Energy controlled insertion of polar molecules in dense fluids. *Journal of Chemical Physics*, 121(24): 12139–12142, 2004.

[130] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2nd edition, 2004.

[131] M. Allen and D. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[132] A. Stone. *The Theory of Intermolecular Forces*. Oxford University Press, 1996.

[133] G. C. Maitland, M. Rigby, E. B. Smith and W. A. Wakeham. *Intermolecular Forces: Their Origin and Determination*. Oxford University Press, 1981.

[134] B. Guillot. A reappraisal of what we have learnt during three decades of computer simulations on water. *Journal of Molecular Liquids*, 101(1-3): 219–260, 2002.

[135] R. Bukowski, K. Szalewicz, G. C. Groenenboom and A. van der Avoird. Predictions of the properties of water from first principles. *Science*, 315(5816): 1249–1252, 2007.

[136] A. J. Stone. Water from first principles. *Science*, 315(5816): 1228–1229, 2007.

[137] W. G. Vincenti and C. H. Kruger. *Introduction to Physical Gas Dynamics*. Wiley, 1965.

[138] J. K. Johnson, J. A. Zollweg and K. E. Gubbins. The Lennard-Jones equation of state revisited. *Molecular Physics*, 78(3): 591–618, 1993.

[139] W. Smith. Molecular dynamics on hypercube parallel computers. *Computer Physics Communications*, 62(2–3): 229–248, 1991.

[140] D. Rapaport. Multi-million particle molecular dynamics. II. Design considerations for distributed processing. *Computer Physics Communications*, 62(2–3): 217–228, 1991.

[141] G. Karypis and V. Kumar. METIS. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Version 4.0. University of Minnesota, http://glaros.dtc.umn.edu/gkhome/views/metis.

[142] J. G. Powles, S. Murad and P. V. Ravi. A new model for permeable micropores. *Chemical Physics Letters*, 188: 21–24, 1992.

[143] H. Goldstein. *Classical Mechanics*. Addison-Wesley, 2nd edition, 1980.

[144] ParaView: Parallel Visualization Application. http://www.paraview.org.

[145] L. D. Landau and E. M. Lifshitz. *Statistical Physics Part 1 (Course of Theoretical Physics v.5)*. Pergamon, 3rd edition, 1980.

[146] T. N. Heinz and P. H. Hunenberger. A fast pairlist-construction algorithm for molecular simulations under periodic boundary conditions. *Journal of Computational Chemistry*, 25(12): 1474 – 1486, 2004.

[147] W. Mattson and B. Rice. Near-neighbor calculations using a modified cell-linked list method. *Computer Physics Communications*, 119(2-3): 135–148, 1999.

[148] P. Gonnet. A simple algorithm to accelerate the computation of non-bonded interactions in cell-based molecular dynamics simulations. *Journal of Computational Chemistry*, 28(2): 570–573, 2007.

[149] K. Fischer, B. Gärtner, T. Herrmann, M. Hoffmann, S. Schnherr and the CGAL Editorial Board. Optimal distances. *CGAL User and Reference Manual*. 3.3 edition, 2007.

[150] W. Gellert, S. Gottwald, M. Hellwich, H. Kästner and H. Künstner (editors). *VNR Concise Encyclopedia of Mathematics*. Van Nostrand Reinhold, New York, 2nd edition, 1989.

[151] E. W. Weisstein. Line-line distance. From MathWorld — A Wolfram Web Resource. http://mathworld.wolfram.com/Line-LineDistance.html.

[152] L. D. Landau and E. M. Lifshitz. *Mechanics (Course of Theoretical Physics v.1)*. Pergamon Press, 3rd edition, 1976.

[153] P. A. N. Nordin. *Complex Chemistry Modeling of Diesel Spray Combustion*. Ph.D. thesis, Chalmers University of Technology, Gothenburg, Sweden, 2000.

[154] X. Nie, S. Chen and M. O. Robbins. Hybrid continuum-atomistic simulation of singular corner flow. *Physics of Fluids*, 16: 3579–3591, 2004.

[155] E. Flekkøy, G. Wagner and J. Feder. Hybrid model for combined particle and continuum dynamics. *Europhysics Letters*, 52(3): 271–276, 2000.

[156] J. Li, D. Liao and S. Yip. Coupling continuum to molecular-dynamics simulation: Reflecting particle method and the field estimator. *Physical Review E*, 57: 7259–7267, 1998.

[157] D. Frenkel and B. Smit. *Understanding Molecular Simulation.* Academic Press, 2001.

[158] C. Braga and K. Travis. A configurational temperature nose-hoover thermostat. *Journal of Chemical Physics*, 123(13): 134101, 2005.

[159] E. A. Koopman and C. P. Lowe. Advantages of a Lowe-Andersen thermostat in molecular dynamics simulations. *Journal of Chemical Physics*, 124(20): 204103, 2006.

[160] M. P. Allen and F. Schmid. A thermostat for molecular dynamics of complex fluids. *Molecular Simulation*, 33(1-2): 21–26, 2007.

[161] K. Travis, D. Searles and D. Evans. Strain rate dependent properties of a simple fluid. *Molecular Physics*, 95(2): 195–202, 1998.

[162] J. Zhang, B. Todd and K. Travis. Viscosity of confined inhomogeneous nonequilibrium fluids. *Journal of Chemical Physics*, 121(21): 10778–86, 2004.

[163] B. Todd, D. Evans and P. Daivis. Pressure tensor for inhomogeneous fluids. *Physical Review E*, 52(2): 1627–1638, 1995.

[164] K. Travis, P. Daivis and D. Evans. Thermostats for molecular fluids undergoing shear flow: application to liquid chlorine. *Journal of Chemical Physics*, 103(24): 10638–51, 1995.

[165] K. Travis, P. Daivis and D. Evans. Computer simulation algorithms for molecules undergoing planar couette flow: a nonequilibrium molecular dynamics study. *Journal of Chemical Physics*, 103(3): 1109–18, 1995.

[166] B. Todd and D. Evans. The heat flux vector for highly inhomogeneous nonequilibrium fluids in very narrow pores. *Journal of Chemical Physics*, 103(22): 9804–9809, 1995.

[167] B. Todd, P. Daivis and D. Evans. Heat flux vector in highly inhomogeneous nonequilibrium fluids. *Physical Review E*, 51(5, pt.A): 4362–4368, 1995.

[168] B. Todd and D. Evans. Temperature profile for poiseuille flow. *Physical Review E*, 55(3, pt.A): 2800–2807, 1997.

[169] A. Murdoch. A critique of atomistic definitions of the stress tensor. *Journal of Elasticity*, 88(2): 113–140, 2007.

[170] A. Murdoch and D. Bedeaux. Continuum equations of balance via weighted averages of microscopic quantities. *Proceedings of The Royal Society of London A*, 445(1923): 157 – 179, 1994.

[171] A. I. Murdoch. *Foundations of Continuum Modelling: a Microscopic Perspective with Applications*. Polish Academy of Sciences, 2003.

[172] J. M. G. Vilar and J. M. Rubi. Thermodynamics "beyond" local equilibrium. *Proceedings of the National Academy of Sciences*, 98(20): 11081–11084, 2001.

[173] J. Powles, G. Rickayzen and D. Heyes. Temperatures: Old, new and middle aged. *Molecular Physics*, 103(10): 1361 – 1373, 2005.

[174] H. Rugh. Dynamical approach to temperature. *Physical Review Letters*, 78(5): 772 – 4, 1997.

[175] H. Rugh. A geometric, dynamical approach to thermodynamics. *Journal of Physics A (Mathematical and General)*, 31(38): 7761 – 70, 1998.

[176] H. H. Rugh. Microthermodynamic formalism. *Physical Review E*, 64(5): 055101, 2001.

[177] D. Hertzog, B. Ivorra, B. Mohammadi, O. Bakajin and J. Santiago. Optimization of a microfluidic mixer for studying protein folding kinetics. *Analytical Chemistry*, 78(13): 4299–4306, 2006.

[178] I. T. Todorov and W. Smith. *The DL_POLY_3 User Manual, version 3.08*. STFC Daresbury Laboratory, 2007.
http://www.cse.scitech.ac.uk/ccg/software/DL_POLY/.

[179] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen. A smooth particle mesh ewald method. *Journal of Chemical Physics*, 103(19): 8577–8593, 1995.

[180] C. Kittel. *Introduction to Solid State Physics*. Wiley, 8th edition, 2005.

[181] D. A. Danielson. *Vectors and Tensors in Engineering and Physics*. Perseus Publishing, 2nd edition, 2000.

[182] R. Varney. Mean free paths, ion drift velocities, and the Poisson distribution. *American Journal of Physics*, 39(5): 534–538, 1971.

[183] D. A. Lockerby, J. M. Reese, D. R. Emerson and R. W. Barber. Velocity boundary condition at solid walls in rarefied gas calculations. *Physical Review E*, 70: 017303, 2004.

[184] D. A. Lockerby, J. M. Reese and M. A. Gallis. Capturing the knudsen layer in continuum-fluid models of nonequilibrium gas flows. *AIAA Journal*, 43(6): 1391–1393, 2005.

[185] The Computer Language Benchmarks Game
http://shootout.alioth.debian.org.

Part IV

Appendices

Appendix A

Reduced unit derivation

Reduced unit expressions for force, time, velocity, acceleration, temperature, pressure and density are derived below in terms of defined values for reference length, σ_r, energy, ϵ_r and mass m_r.

Force

From energy = **force** $\cdot$ **distance**:

$$f = \frac{\epsilon}{l} = \frac{\epsilon^* \epsilon_r}{l^* \sigma_r},$$
$$f\left(\frac{\sigma_r}{\epsilon_r}\right) = \frac{\epsilon^*}{l^*},$$

therefore,

$$f^* = f\left(\frac{\sigma_r}{\epsilon_r}\right). \tag{A.1}$$

Time

From Newton's second law:

$$f = ma = \frac{ml}{t^2},$$
$$f^*\left(\frac{\epsilon_r}{\sigma_r}\right) = \frac{m^* m_r l^* \sigma_r}{t^2},$$
$$f^* = \frac{m^* l^*}{t^2}\left(\frac{m_r \sigma_r^2}{\epsilon_r}\right),$$

therefore

$$t^* = t\sqrt{\frac{\epsilon_r}{m_r\sigma_r^2}}. \tag{A.2}$$

Velocity

Manipulating the magnitude of velocity only, using speed = distance / time:

$$\begin{aligned} v &= \frac{l}{t}, \\ &= \frac{l^*\sigma_r}{t^*\sqrt{m_r\sigma_r^2/\epsilon_r}}, \\ &= \frac{l^*}{t^*}\sqrt{\frac{\epsilon_r}{m_r}}, \\ v\sqrt{\frac{m_r}{\epsilon_r}} &= \frac{l^*}{t^*}, \end{aligned}$$

therefore

$$v^* = v\sqrt{\frac{m_r}{\epsilon_r}}. \tag{A.3}$$

Acceleration

Manipulating the magnitude only, using acceleration = velocity / time:

$$\begin{aligned} a &= \frac{v}{t}, \\ &= \frac{v^*\sqrt{\epsilon_r/m_r}}{t^*\sqrt{m_r\sigma_r^2/\epsilon_r}}, \\ &= \frac{v^*}{t^*}\left(\frac{\epsilon_r}{m_r\sigma_r}\right), \\ a\left(\frac{m_r\sigma_r}{\epsilon_r}\right) &= \frac{v^*}{t^*}, \end{aligned}$$

therefore,

$$a^* = a\left(\frac{m_r\sigma_r}{\epsilon_r}\right). \tag{A.4}$$

Temperature

Using the equipartition definition of temperature (where k_b is the Boltzmann constant):

$$\begin{aligned} T &= \frac{2}{3k_bN}\sum_N \frac{1}{2}mv^2, \\ &= \frac{2}{3k_bN}\sum_N \frac{1}{2}m^*m_rv^{*2}\left(\frac{\epsilon_r}{m_r}\right), \\ &= \left(\frac{\epsilon_r}{k_b}\right)\frac{2}{3N}\sum_N \frac{1}{2}m^*v^{*2}, \\ T\left(\frac{k_b}{\epsilon_r}\right) &= \frac{2}{3N}\sum_N \frac{1}{2}m^*v^{*2}, \end{aligned}$$

therefore,

$$T^* = T\left(\frac{k_b}{\epsilon_r}\right). \tag{A.5}$$

Pressure

From pressure = force / area:

$$\begin{aligned} P &= \frac{f}{l^2}, \\ &= \frac{f^*\left(\epsilon_r/\sigma_r\right)}{l^{*2}\sigma_r^2}, \\ P\left(\frac{\sigma_r^3}{\epsilon_r}\right) &= \frac{f^*}{l^{*2}}, \end{aligned}$$

therefore

$$P^* = P\left(\frac{\sigma_r^3}{\epsilon_r}\right). \tag{A.6}$$

Mass density

Mass density is defined as total mass divided by volume:

$$\begin{aligned} \rho_M &= \frac{M}{V}, \\ &= \frac{M^*}{V^*}\left(\frac{m_r}{\sigma_r^3}\right), \\ \rho_M\left(\frac{\sigma_r^3}{m_r}\right) &= \frac{M^*}{V^*}, \end{aligned}$$

therefore

$$\rho_M^* = \rho_M\left(\frac{\sigma_r^3}{m_r}\right). \tag{A.7}$$

Number density

Number density is defined as total number (which has no unit) divided by volume:

$$\begin{aligned} \rho_N &= \frac{N}{V}, \\ &= \frac{N}{V^*\sigma_r^3}, \\ \rho_N\sigma_r^3 &= \frac{N}{V^*}, \end{aligned}$$

therefore

$$\rho_N^* = \rho_N\sigma_r^3. \tag{A.8}$$

Appendix B

Derivation of shifted force equation

The equation for the intermolecular force between molecules interacting with a shifted force Lennard-Jones potential (for the case where $r_{ij} < r_{cut}$) is derived by substituting equation (5.7) into equation (5.5):

$$\mathbf{f}_{ij} = -\nabla\left(4\epsilon\left[\left(\left(\frac{r_{ij}}{\sigma}\right)^{-12} - \left(\frac{r_{ij}}{\sigma}\right)^{-6}\right) - \left(\left(\frac{r_{cut}}{\sigma}\right)^{-12} - \left(\frac{r_{cut}}{\sigma}\right)^{-6}\right) + \frac{12r_{cut}\left(r_{ij} - r_{cut}\right)}{\sigma^2}\left(\left(\frac{r_{cut}}{\sigma}\right)^{-14} - \frac{1}{2}\left(\frac{r_{cut}}{\sigma}\right)^{-8}\right)\right]\right). \quad \text{(B.1)}$$

The values of σ, r_{cut} and ϵ are constant, so

$$\nabla\left(\left(\frac{r_{cut}}{\sigma}\right)^{-12} - \left(\frac{r_{cut}}{\sigma}\right)^{-6}\right) = 0,$$

$$\nabla\left(r_{cut}\left(\left(\frac{r_{cut}}{\sigma}\right)^{-14} - \frac{1}{2}\left(\frac{r_{cut}}{\sigma}\right)^{-8}\right)\right) = 0,$$

and the following constant is defined for convenience:

$$C_{SF} = \frac{12r_{cut}}{\sigma^2}\left(\left(\frac{r_{cut}}{\sigma}\right)^{-14} - \frac{1}{2}\left(\frac{r_{cut}}{\sigma}\right)^{-8}\right).$$

Equation (B.1) then becomes:

$$\mathbf{f}_{ij} = -\nabla\left(4\epsilon\left[\left(\frac{r_{ij}}{\sigma}\right)^{-12} - \left(\frac{r_{ij}}{\sigma}\right)^{-6} + C_{SF}r_{ij}\right]\right). \quad \text{(B.2)}$$

Figure 5.1 shows the sign convention for the force vector. $\mathbf{f}_{ij}$, the force acting on molecule i due to j, is repulsive (points away from the pair of molecules in the direction of $\mathbf{r}_{ij}$) when $-\nabla U(r_{ij})$ is positive, i.e. the portion of the potential where U decreases with increasing r_{ij}. Using summation notation [181] where $\mathbf{e}_k$ is a unit vector in the direction r_k ($k = \{x, y, z\}$ for a Cartesian coordinate system),

$$\mathbf{f}_{ij} = -\mathbf{e}_k \frac{\partial}{\partial r_k} \left(4\epsilon \left[\left(\frac{r_{ij}}{\sigma} \right)^{-12} - \left(\frac{r_{ij}}{\sigma} \right)^{-6} + C_{SF} r_{ij} \right] \right), \tag{B.3}$$

where,

$$\begin{aligned} \frac{\partial r_{ij}}{\partial r_k} &= \frac{\partial}{\partial r_k} \left(r_x^2 + r_y^2 + r_z^2 \right)^{\frac{1}{2}}, \\ &= \frac{1}{2} \left(r_x^2 + r_y^2 + r_z^2 \right)^{-\frac{1}{2}} 2r_k, \\ &= \frac{r_k}{r_{ij}} \end{aligned}$$

is required to apply the chain rule to equation (B.3). Also

$$\nabla r_{ij} = \mathbf{e}_k \frac{\partial r_{ij}}{\partial r_k} = \mathbf{e}_k \frac{r_k}{r_{ij}} = \frac{\mathbf{r}_{ij}}{r_{ij}} = \hat{\mathbf{r}}_{ij},$$

where $\hat{\mathbf{r}}_{ij}$ is the unit vector in the direction $\mathbf{r}_{ij}$. Equation (B.3) then becomes

$$\begin{aligned} \mathbf{f}_{ij} &= -4\epsilon \mathbf{e}_k \frac{\partial}{\partial r_k} \left(\left(\frac{r_{ij}}{\sigma} \right)^{-12} - \left(\frac{r_{ij}}{\sigma} \right)^{-6} \right) - 4\epsilon C_{SF} \hat{\mathbf{r}}_{ij}, \\ &= -4\epsilon \mathbf{e}_k \left(-\frac{12 r_k}{r_{ij}\sigma} \left(\frac{r_{ij}}{\sigma} \right)^{-13} + \frac{6 r_k}{r_{ij}\sigma} \left(\frac{r_{ij}}{\sigma} \right)^{-7} \right) - 4\epsilon C_{SF} \hat{\mathbf{r}}_{ij}, \\ &= 4\epsilon r_k \mathbf{e}_k \left(\frac{12}{\sigma^2} \left(\frac{r_{ij}}{\sigma} \right)^{-14} - \frac{6}{\sigma^2} \left(\frac{r_{ij}}{\sigma} \right)^{-8} \right) - 4\epsilon C_{SF} \hat{\mathbf{r}}_{ij}, \\ &= 4\epsilon \left(\frac{12}{\sigma^2} \left(\frac{r_{ij}}{\sigma} \right)^{-14} - \frac{6}{\sigma^2} \left(\frac{r_{ij}}{\sigma} \right)^{-8} \right) \mathbf{r}_{ij} - 4\epsilon C_{SF} \hat{\mathbf{r}}_{ij}. \end{aligned}$$

Reintroducing C_{SF} to give the full expression:

$$\begin{aligned} \mathbf{f}_{ij} &= 4\epsilon \left(\frac{12}{\sigma^2} \left(\frac{r_{ij}}{\sigma} \right)^{-14} - \frac{6}{\sigma^2} \left(\frac{r_{ij}}{\sigma} \right)^{-8} \right) \mathbf{r}_{ij} - \frac{48\epsilon r_{cut}}{\sigma^2} \left(\left(\frac{r_{cut}}{\sigma} \right)^{-14} - \frac{1}{2} \left(\frac{r_{cut}}{\sigma} \right)^{-8} \right) \hat{\mathbf{r}}_{ij}, \\ &= \frac{48\epsilon}{\sigma^2} \left(\left(\frac{r_{ij}}{\sigma} \right)^{-14} - \frac{1}{2} \left(\frac{r_{ij}}{\sigma} \right)^{-8} - \frac{r_{cut}}{r_{ij}} \left(\left(\frac{r_{cut}}{\sigma} \right)^{-14} - \frac{1}{2} \left(\frac{r_{cut}}{\sigma} \right)^{-8} \right) \right) \mathbf{r}_{ij}. \end{aligned} \tag{B.4}$$

Rewriting equation (B.4) in reduced units (see section 5.1.1 and appendix A), but retaining ϵ and σ throughout to keep the expression general and allow simulations containing multiple species with different potentials. Substituting $\sigma = \sigma^*\sigma_r$, $r = r^*\sigma_r$ and $\epsilon = \epsilon^*\epsilon_r$ into equation (B.4):

$$\mathbf{f}_{ij} = \frac{48\epsilon^*\epsilon_r}{\sigma^{*2}\sigma_r^2}\left(\left(\frac{r^*_{ij}\sigma_r}{\sigma^*\sigma_r}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{ij}\sigma_r}{\sigma^*\sigma_r}\right)^{-8} - \frac{r^*_{cut}\sigma_r}{r^*_{ij}\sigma_r}\left(\left(\frac{r^*_{cut}\sigma_r}{\sigma^*\sigma_r}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{cut}\sigma_r}{\sigma^*\sigma_r}\right)^{-8}\right)\right)\mathbf{r}^*_{ij}\sigma_r,$$

which becomes,

$$\mathbf{f}^*_{ij} = \mathbf{f}_{ij}\left(\frac{\sigma_r}{\epsilon_r}\right) = \frac{48\epsilon^*}{\sigma^{*2}}\left(\left(\frac{r^*_{ij}}{\sigma^*}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{ij}}{\sigma^*}\right)^{-8} - \frac{r^*_{cut}}{r^*_{ij}}\left(\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-8}\right)\right)\mathbf{r}^*_{ij}. \quad \text{(B.5)}$$

Substituting reduced unit quantities into equation (5.7) (for $r_{ij} < r_{cut}$) and cancelling as above gives the reduced unit expression for the corresponding intermolecular energy:

$$U^*_{SF} = \frac{U_{SF}}{\epsilon_r} = 4\epsilon^*\left[\left(\left(\frac{r^*_{ij}}{\sigma^*}\right)^{-12} - \left(\frac{r^*_{ij}}{\sigma^*}\right)^{-6}\right) - \left(\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-12} - \left(\frac{r^*_{cut}}{\sigma^*}\right)^{-6}\right) + \frac{12r^*_{cut}\left(r^*_{ij} - r^*_{cut}\right)}{\sigma^{*2}}\left(\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-8}\right)\right]. \quad \text{(B.6)}$$

At the start of the simulation, the terms

$$\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-12} - \left(\frac{r^*_{cut}}{\sigma^*}\right)^{-6}, \quad \text{(B.7)}$$

and

$$r^*_{cut}\left(\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-14} - \frac{1}{2}\left(\frac{r^*_{cut}}{\sigma^*}\right)^{-8}\right), \quad \text{(B.8)}$$

are calculated and stored to be used during every force evaluation because they are constant. The term (B.7) can be used in equation (B.6). The term (B.8) can be used in equations (B.5) and (B.6).

A similar process to the above will be required to derive the equations for intermolecular forces when a new class of potential is added to the code.

Appendix C

Neighbour list lifetime prediction

The computational cost of the neighbour list algorithm depends on the number of timesteps a neighbour list remains valid for — a quantity whose dependence on simulation properties can be predicted. For a given number of molecules, N, of mass m, in equilibrium at a temperature T, there is a probability of $1/N$ that a molecule in the system will be travelling with a velocity v_N. It is therefore likely that at every timestep there will be *one* molecule travelling at, or close to this velocity. Given that a neighbour list is invalidated when the cumulative sum of *maximum* displacements in the system exceeds a fixed threshold (usually $0.5\Delta R$, i.e. half the user defined guard radius [130]), finding the number of timesteps required for a molecule travelling at v_N to cover this distance gives an estimate of the average lifetime of the list. Estimating v_N by equating the Maxwellian velocity distribution to $1/N$, where T, m and v_N are in reduced MD units (see figure C.1):

$$4\pi \left(\frac{m}{2\pi T}\right)^{\frac{3}{2}} v_N^2 e^{\left(\frac{-mv_N^2}{2T}\right)} = \frac{1}{N}. \tag{C.1}$$

The high speed solution is,

$$v_N(T, m, N) = \sqrt{-\frac{2T}{m} W_{-1}\left(-\frac{1}{4N}\sqrt{\frac{2\pi T}{m}}\right)}, \tag{C.2}$$

where W_{-1} is the secondary real branch of the Lambert W function.

The accuracy of v_N as a prediction of the maximum velocity in the system has been tested experimentally. Two systems containing 1728 and 13824 Lennard-Jones molecules, $m = 1$, at a number density of 0.8 were equilibrated to two

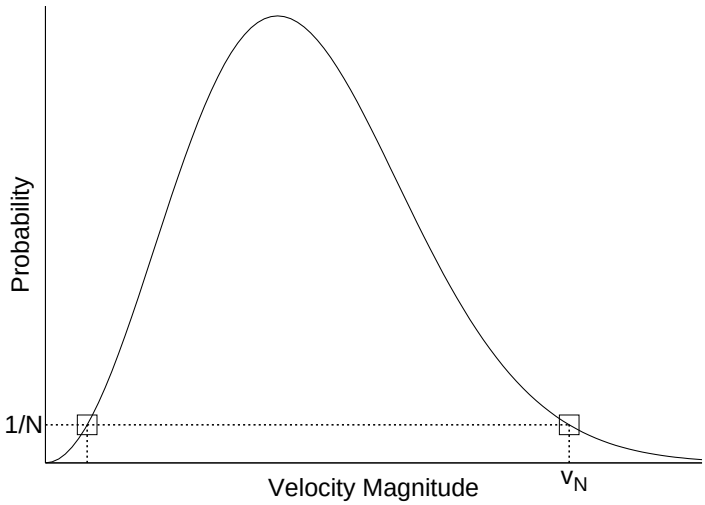


Figure C.1: Maxwellian velocity distribution. Finding velocities corresponding to a probability of $1/N$, where N is the number of molecules in the system. There are two real, positive solutions — the high speed one is v_N, the quantity of interest.

different temperatures, $T = 1.0$ and $T = 2.5$, then simulated for 44000 timesteps of $\Delta t = 0.005$. At each timestep the maximum velocity in the system was found and added to a histogram with a velocity bin width of 0.05. The histograms were then normalised to produce probability functions, their mean calculated, and the data was fitted to a curve of the form,

$$f_{v_N}(x) = p(x - s)^q e^{-(x-s)/r}. \tag{C.3}$$

Figure C.2 shows the maximum velocity probability functions with their mean and the corresponding value of $v_N(T, m, N)$. Table C.1 shows the pertinent data and the curve fitting parameters for each result. The values of v_N and the distribution mean are close in all four cases, with v_N consistently higher (2.5% – 5.5% in these examples) but lying comfortably within the distribution. Using v_N will result in a slightly pessimistic estimate of the lifetime of a neighbour list.

Table C.1: Experimental data and curve fitting parameters for $f_{v_N}(x)$.

T	N	v_N	mean	$\frac{v_N - \text{mean}}{\text{mean}}$	p	q	r	s
1	1728	4.528	4.294	5.45%	17310	6.794	0.1008	3.504
1	13824	5.006	4.783	4.66%	47060	7.027	0.08975	4.058
2.5	1728	6.979	6.821	2.31%	610.7	8.334	0.1494	5.416
2.5	13824	7.756	7.558	2.62%	1276	7.157	0.1406	6.406

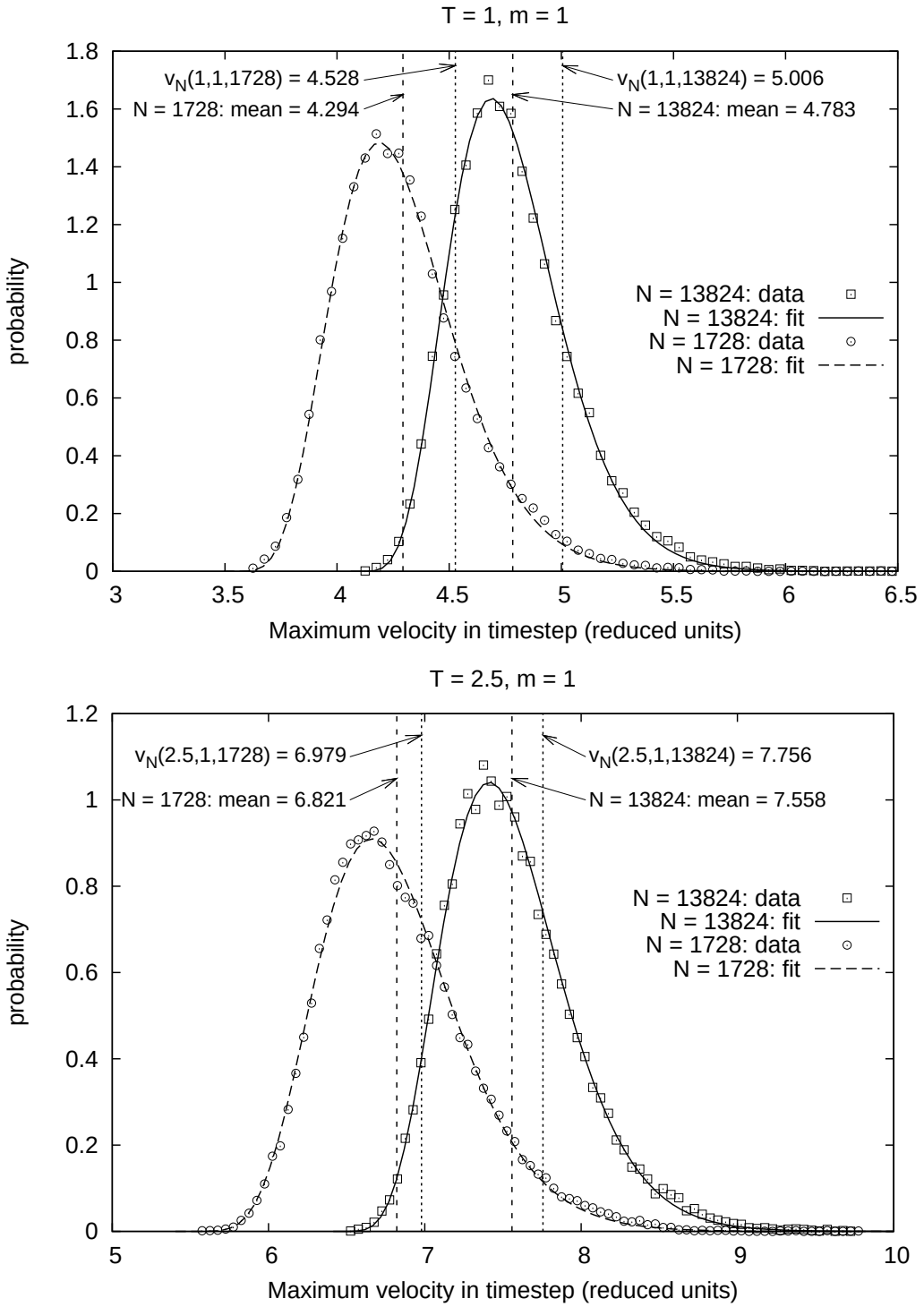


Figure C.2: Probability of maximum velocity in simulation at $T = 1.0$ (top) and $T = 2.5$ (bottom) for 1728 and 13824 Lennard-Jones molecules of mass $m = 1$ compared to $v_N(T, m, N)$. The data was collected from 44000 timesteps of $\Delta t = 0.005$. The mean of the distributions is shown for comparison. Temperature, velocity and mass are expressed in reduced units.

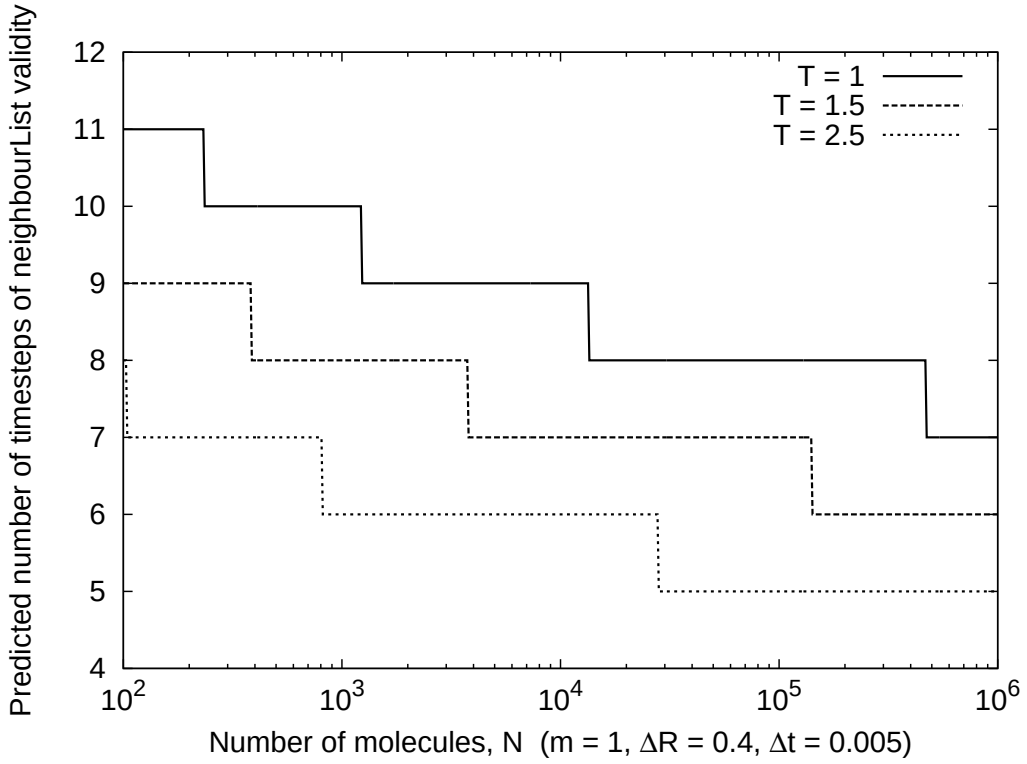


Figure C.3: Variation of estimated neighbour list lifetime with typical simulation parameters for a Lennard-Jones fluid [130]. $T, m, \Delta R$ and Δt are in reduced MD units.

The number of timesteps (of length Δt) a neighbour list is valid for, L, is given by

$$L = \left\lceil \frac{0.5\Delta R}{\Delta t v_N} \right\rceil . \tag{C.4}$$

This lifetime reduces with increasing temperature and number of molecules, as shown in figure C.3, resulting in a higher computational cost when using neighbour lists. At higher temperatures, v_N is higher because the velocity distribution covers a higher molecular speed range. In systems comprising more molecules, it is more probable that at least one molecule will be travelling at a given speed in the upper region of the distribution, therefore v_N is also higher. The cost of an algorithm based only on cell interactions is not sensitive to either of these parameters, therefore becomes increasingly attractive in large systems ($> 10^6$ molecules) where neighbour list lifetimes are lower. The use of the ceiling function in equation (C.4) reduces the impact of the difference between v_N and the measured mean of the maximum velocity probability function, resulting only in an alteration of the position of the 'steps' in the result.

Appendix D

Build cell interactions

Building DILs, creating referred cells and determining which real cells are in range of them is allowed to be computationally expensive (within reason) because it will only happen once at the beginning of the MD simulation, if the mesh is static. Accessing the information, however, must be as fast as possible because it will happen at every timestep. In each of these steps either the PP, PPGR or PFEE method can be used to test whether or not two cells are within r_{cut} of each other.

D.1 Building DILs

The construction of the DILs and accessing molecules is done in such a way as to not double-count interactions. For example, if cells A and B need to interact, cell B will be on cell A's DIL, but cell A will not be on cell B's DIL. When a molecule in cell A interacts with one in cell B, the molecule in cell B will receive the inverse of the force vector calculated to be added to the molecule in cell A, because the pair forces are reciprocal, i.e. $\mathbf{f}_{ab} = -\mathbf{f}_{ba}$, because of Newton's third law.

When using PP or PPGR, all points in the mesh are compared, as per algorithm 1 (page 66). For each step in each method, two sets of cells are returned. When two points are in range, this produces a set of cells attached to one point and another set attached to the other point. When using PFEE, all points are compared to all faces. When a point-face pair are in range, the set of cells that are attached to the point and the set of cells that the face forms part of are produced. All edges in the mesh are compared in a non-double-counting loop, and when they are in range two sets of cells are again produced — those attached to

one edge and those attached to the other. The index of each cell in one set is compared to the index of each cell in the other set, and the cell with the higher index is added to the DIL of the cell with the lower index. When the indices are equal, cells are not added to their own DIL.

D.2 Creating referred cells

Coupled patches (see section 5.4.1) are the basis of periodic and interprocessor communication for creating referred cells. Coupled patches provide two surfaces; a molecule which exits one enters the other. Two types are used in AICA:

- *Periodic patches* on a single processor are arranged into two halves, each half representing one of the coupled periodic surfaces. When a molecule crosses a face on one surface, it is wrapped around to the corresponding position on the corresponding face on the other surface.
- *Processor patches* provide links between portions of the mesh on different processors, one half of the patch is on each processor. When a molecule crosses a face on one surface, it is moved to the corresponding position on the corresponding face on the other surface, on the other processor. Decomposing a mesh for parallel processing will often require a periodic patch to be changed to a processor patch.

The surfaces of coupled patches may have any relative orientation, and may be spatially separated as long as the face pairs on each surface correspond to each other.

D.2.1 Creating patch segments

In the decomposed portion of the mesh on each processor, each processor and periodic patch should be split into segments, such that:

- faces on a processor patch that were internal to the mesh prior to decomposition end up on a segment;
- faces on a processor patch that were on separate periodic patches in the undecomposed mesh end up on different segments. These segments are further split such that faces that were on different halves of the periodic patch on the undecomposed mesh end up on different segments;

- faces on different halves of a periodic patch end up on different segments.

Each segment must produce a single vector/tensor transformation pair (see appendix E) which will be applied to all cells referred across it.

D.2.2 Cell referring iterations

For each patch segment:

1. Find all real and existing referred cells with a portion of their surface in range of the faces comprising the patch segment.

2. Refer or re-refer this set of cells across the boundary defined by the patch segment using its transformation, see appendix E. In the case of a segment of periodic boundary, this creates new referred cells on the same processor. For a segment of a processor patch, these cells are communicated to, and created on the neighbouring processor. Before creating any new referred cell a check is carried out to ensure that it is not

   - a duplicate referred cell, one that has been created already by being referred across a different segment;
   - a referred cell trying to be duplicated on top of a real cell, i.e. a cell being referred back on top of itself.

   If the proposed cell for referral would create a duplicate of an existing referred cell, or end up on top of a real cell, then it is not created. To be a duplicate, the source processor, source cell and the vector part of the transformation (see appendix E) must be the same for the two cells (note, the vector part of the transformation for a real cell is zero by definition).

A single run of these steps will usually not produce all of the required referred cells. They are repeated until no processor adds a referred cell in a complete evaluation of all segments, meaning all possible interactions are accounted for. In iterations after the first, in step 1 it is enough to search only for referred cells in range of the faces on the patch segment, because the real cells will not have changed and would all be duplicates. The final configuration of referred cells does not depend on the order of patch segment evaluation.

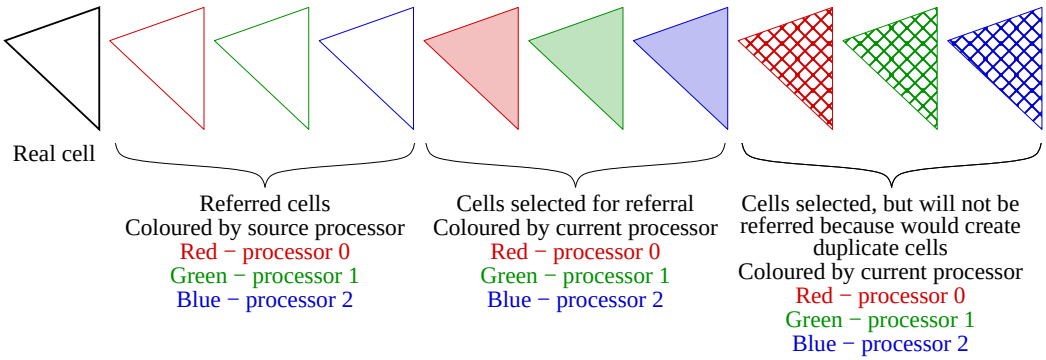


Figure D.1: Key to colour coding of cells in example construction of referred cells.

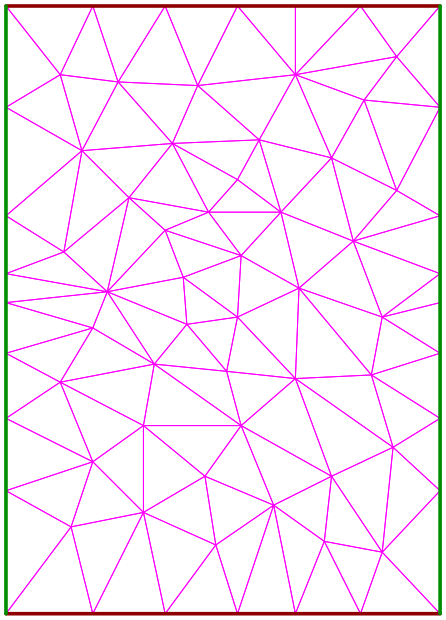
D.3 Determining real cells in range of referred cells

Once all of the referred cells have been created, each referred cell searches the mesh to determine which real cells have part of their surface within r_{cut} range of the surface of the referred cell, and therefore need to be supplied with referred molecule interactions. The referred cell stores which real cells it needs to interact with. This process is accelerated by only searching a subset of the real cells in the mesh. These are the real cells that were identified as being in range of any processor or periodic patch when creating the referred cells. This set is guaranteed to contain all cells that could be in range of a referred cell.

D.4 Example construction of referred cells

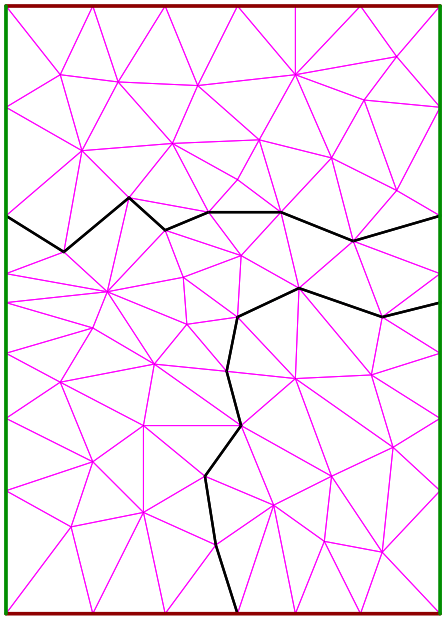
The figures on pages 210 to 227 show the referred cell construction algorithm operating on a mesh that has been decomposed to run in parallel. The example is in 2D for clarity but the process is exactly the same in 3D. In this example the number of referred cells created exceeds the number of real cells, which would lead to much costly interprocessor communication. This is because the mesh has been made deliberately small relative to r_{cut} to demonstrate as many features of the algorithm as possible, and to be understandable. In realistic systems, the mesh portions would be significantly bigger than r_{cut} and the referred cells would form a relatively thin 'halo' or 'skin' around each portion. Decomposition of the mesh should preferably be carried out to minimise the number of cells that need to be referred and to ensure that the vast majority of the intermolecular interactions

happen between real-real molecule pairs; in this way the communication cost is minimised.

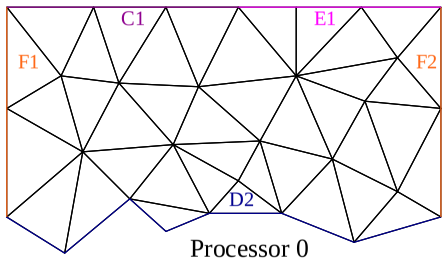


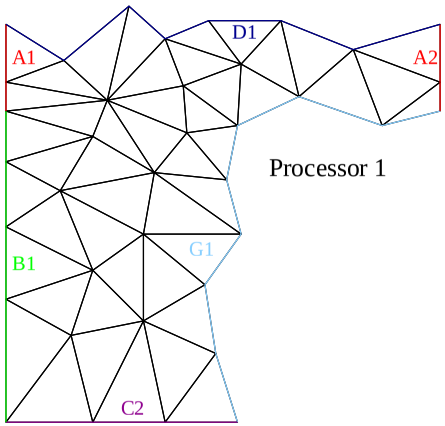
Initial mesh

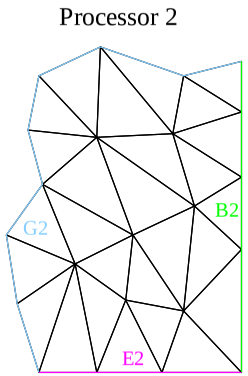
Rectangular domain containing triangular cells. The domain is periodic top-bottom and left-right.



Select partitioning:
Decomposing the domain for calculation on 3 processors. Decomposition chosen such that non-neighbouring processors (top and lower right) must interact.



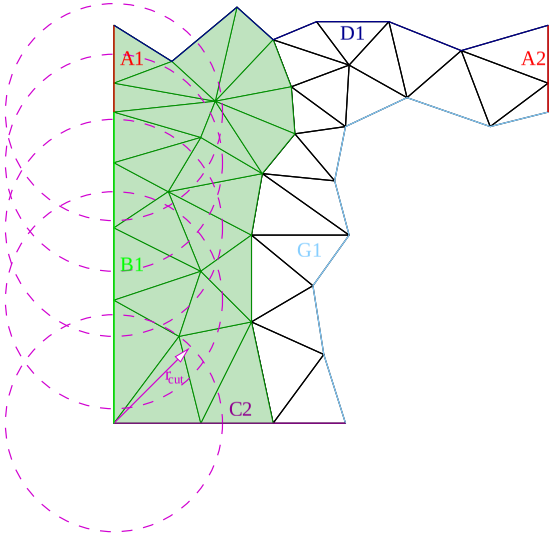


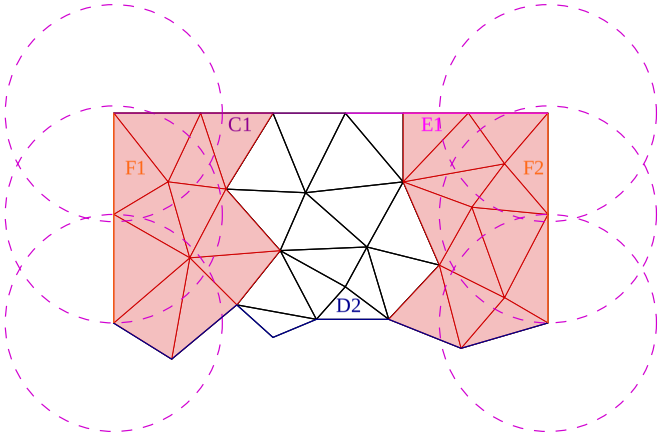


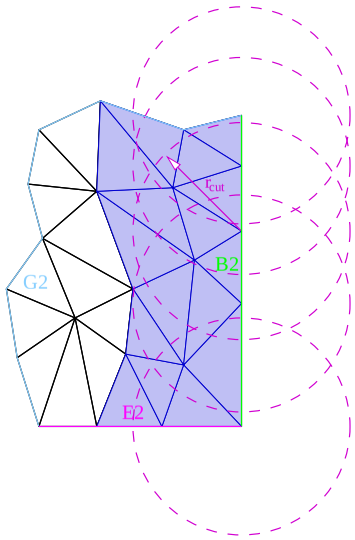
Identify patch segments.

Mesh segments separated for clarity, coordinates are still continuous across the boundaries.

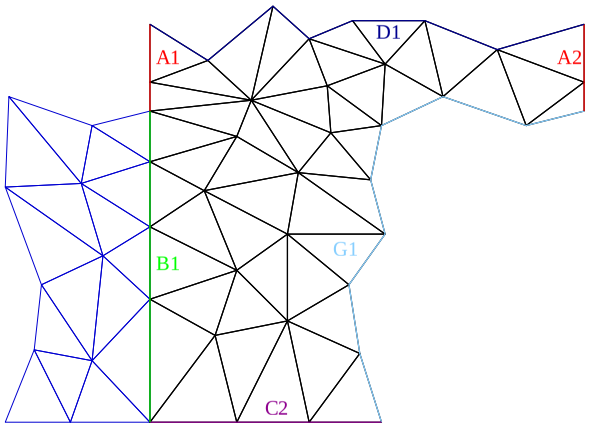
Split processor and periodic patches into segments, A-G. Arbitrary order of segment evaluation, B,F,A,D,G,C,E chosen. Processor and periodic boundaries (A and F) are treated in exactly the same way.

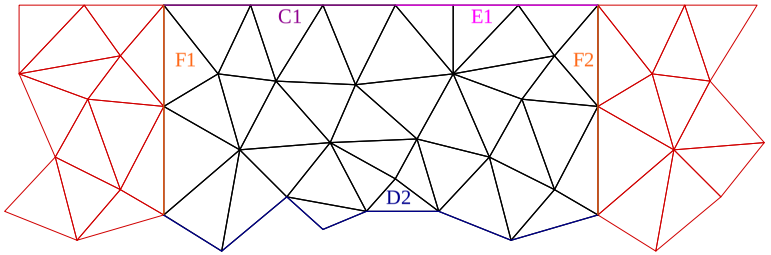


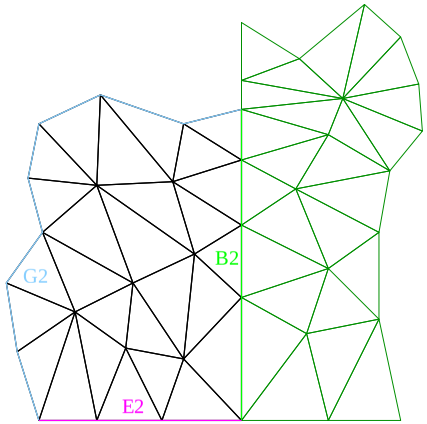




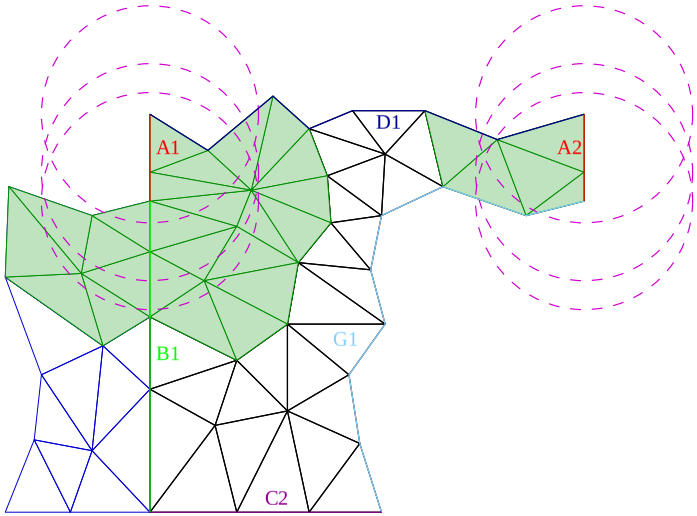
Segment B and F: Select cells.
All cells within r_{cut} (shown approximately by circles drawn from vertices on segments) of a face on segment B1 or B2 selected for referral, similarly for F1 and F2. See figure D.1 for cell colour coding.

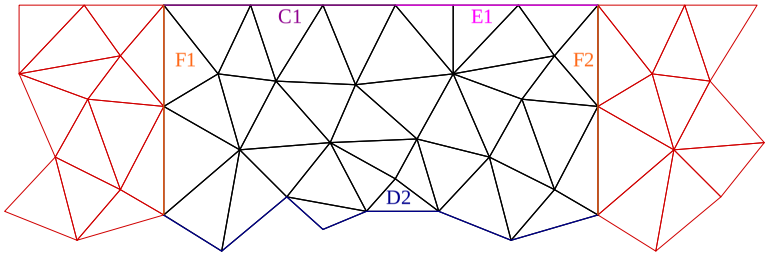


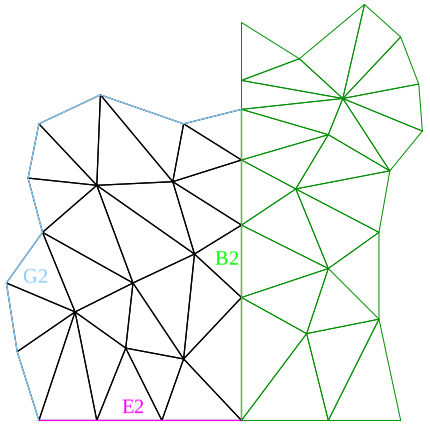




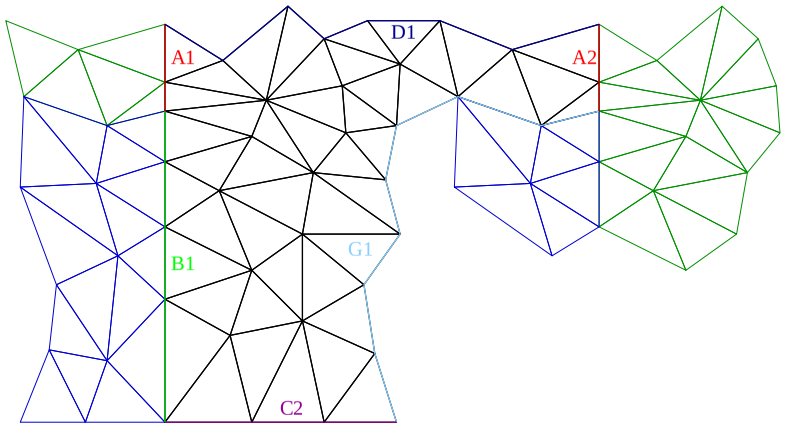
Segment B and F: Refer cells.
Selected cells referred across segments B and F, see figure D.1 for cell colour coding.

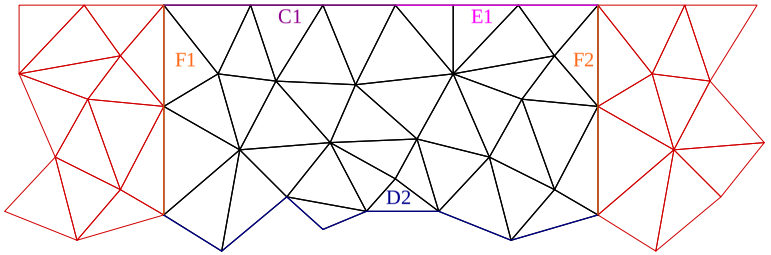


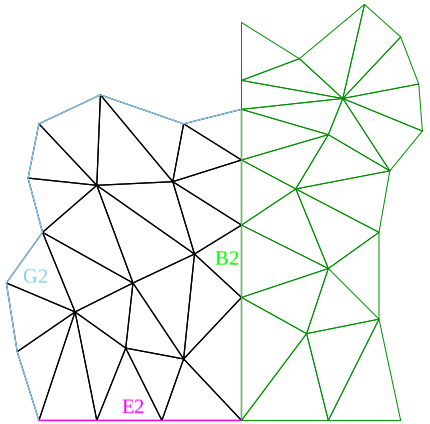




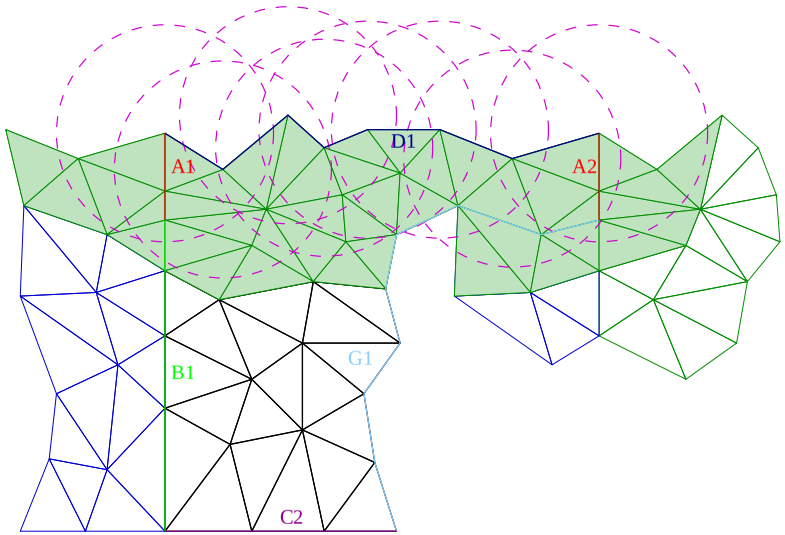
Segment A: Select cells.
All real and existing referred cells within r_{cut} (shown approximately by circles drawn from vertices on segments) of a face on segment A1 or A2 selected for referral. Cells referred across segment B are being rereferred by segment A1.

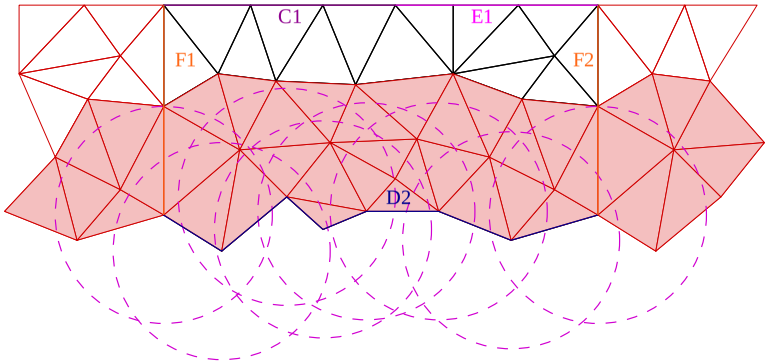


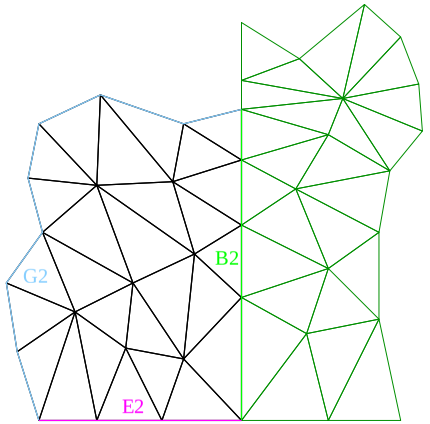




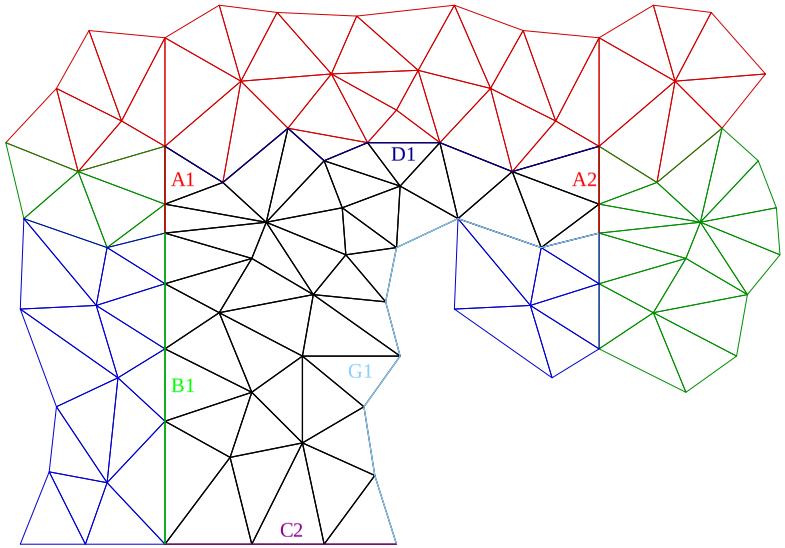
Segment A: Refer cells.

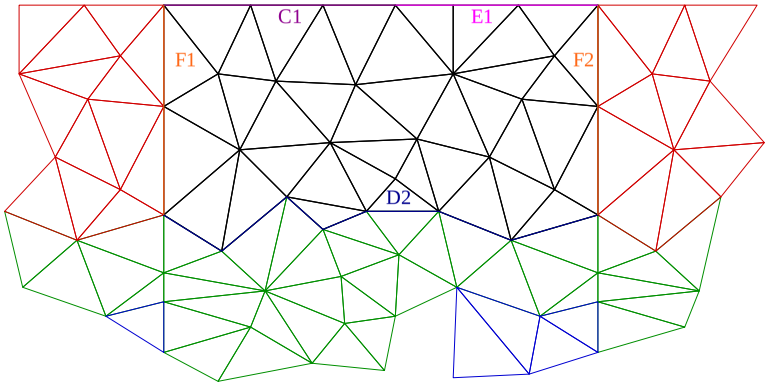


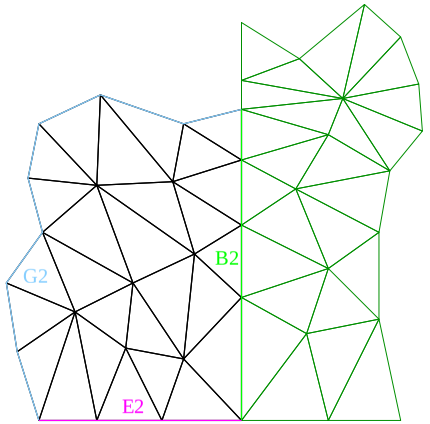




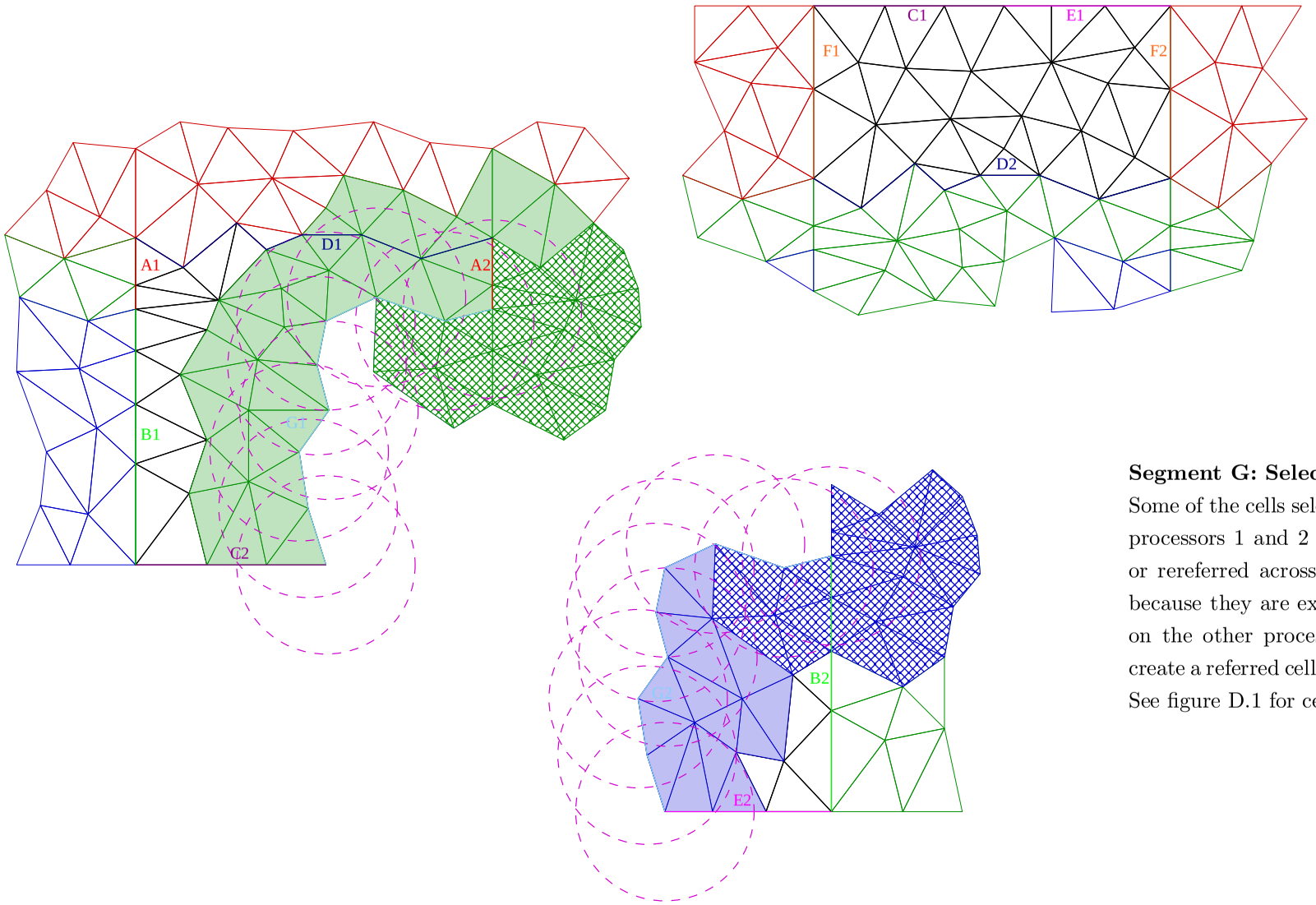
Segment D: Select cells.



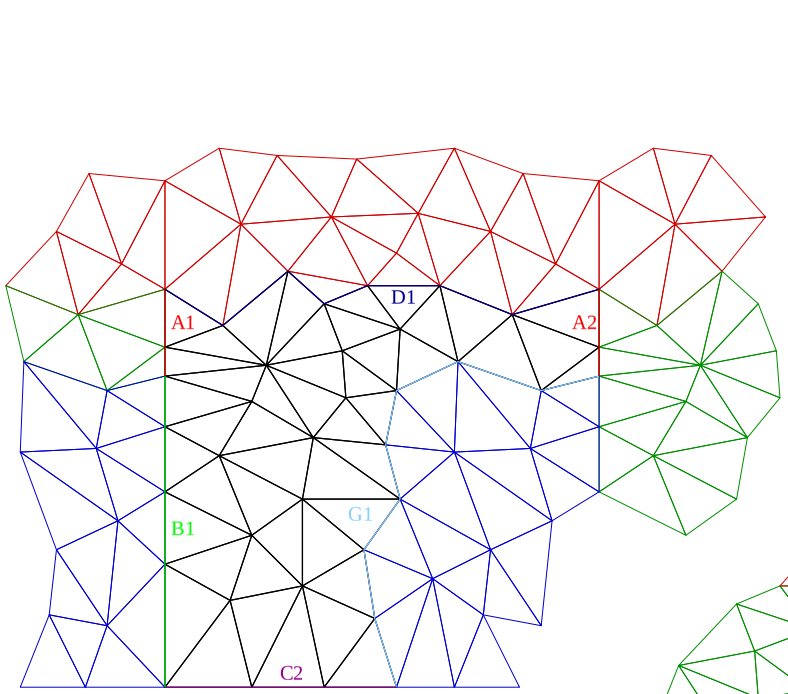


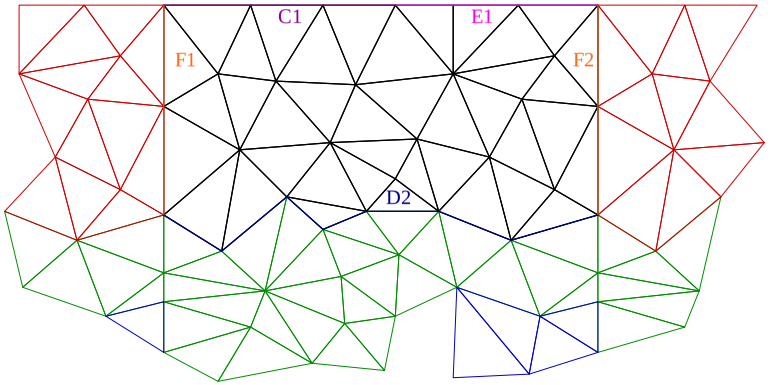


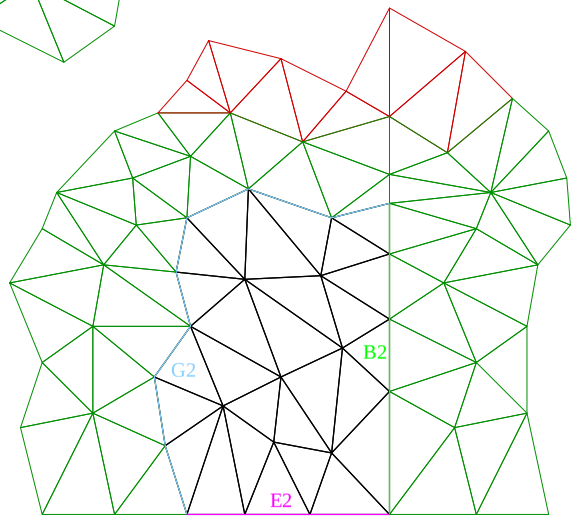
Segment D: Refer cells.



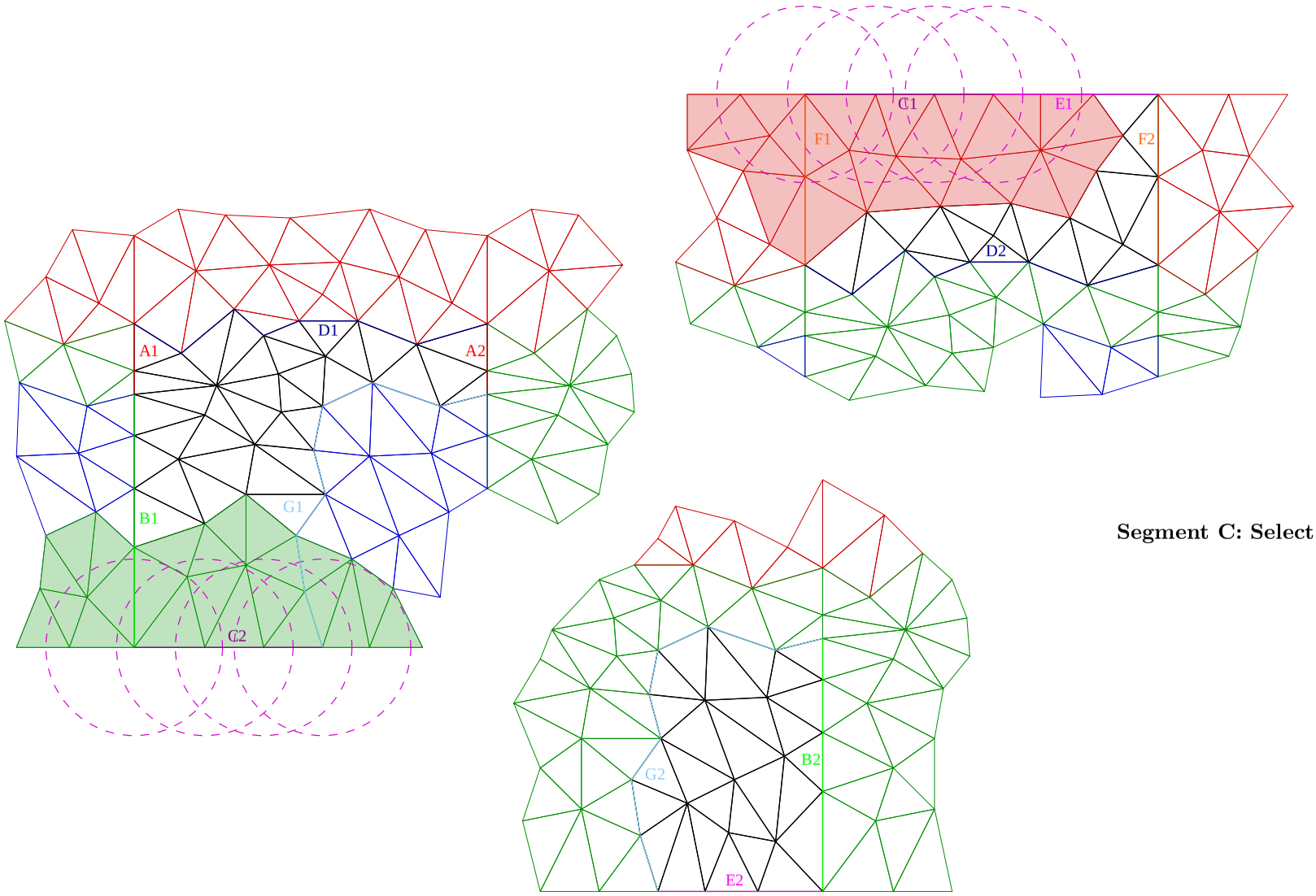
Segment G: Select cells.
Some of the cells selected for referral on processors 1 and 2 do not get referred or rereferred across segment G, either because they are existing referred cells on the other processor or they would create a referred cell on top of a real cell. See figure D.1 for cell colour coding.



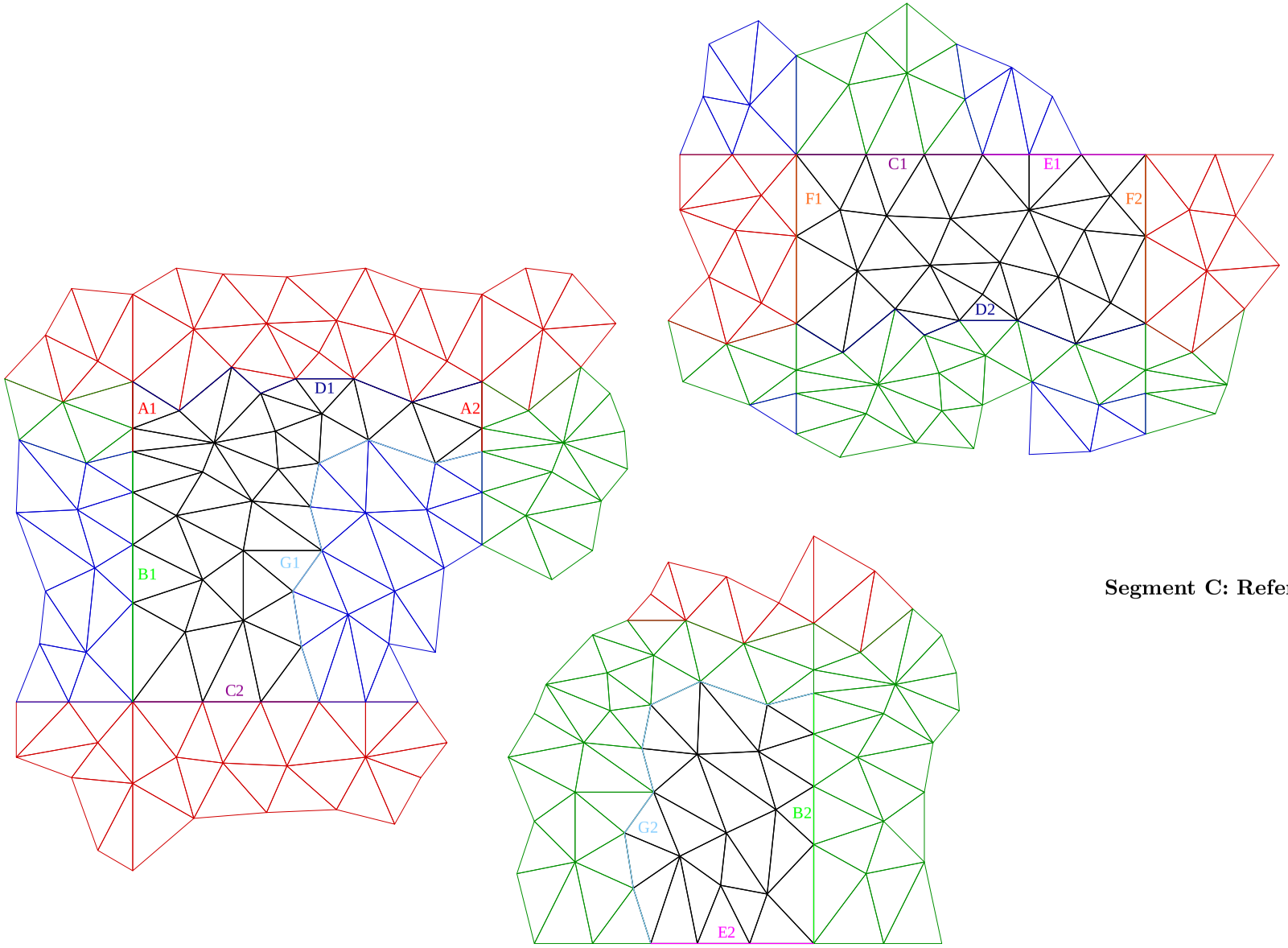




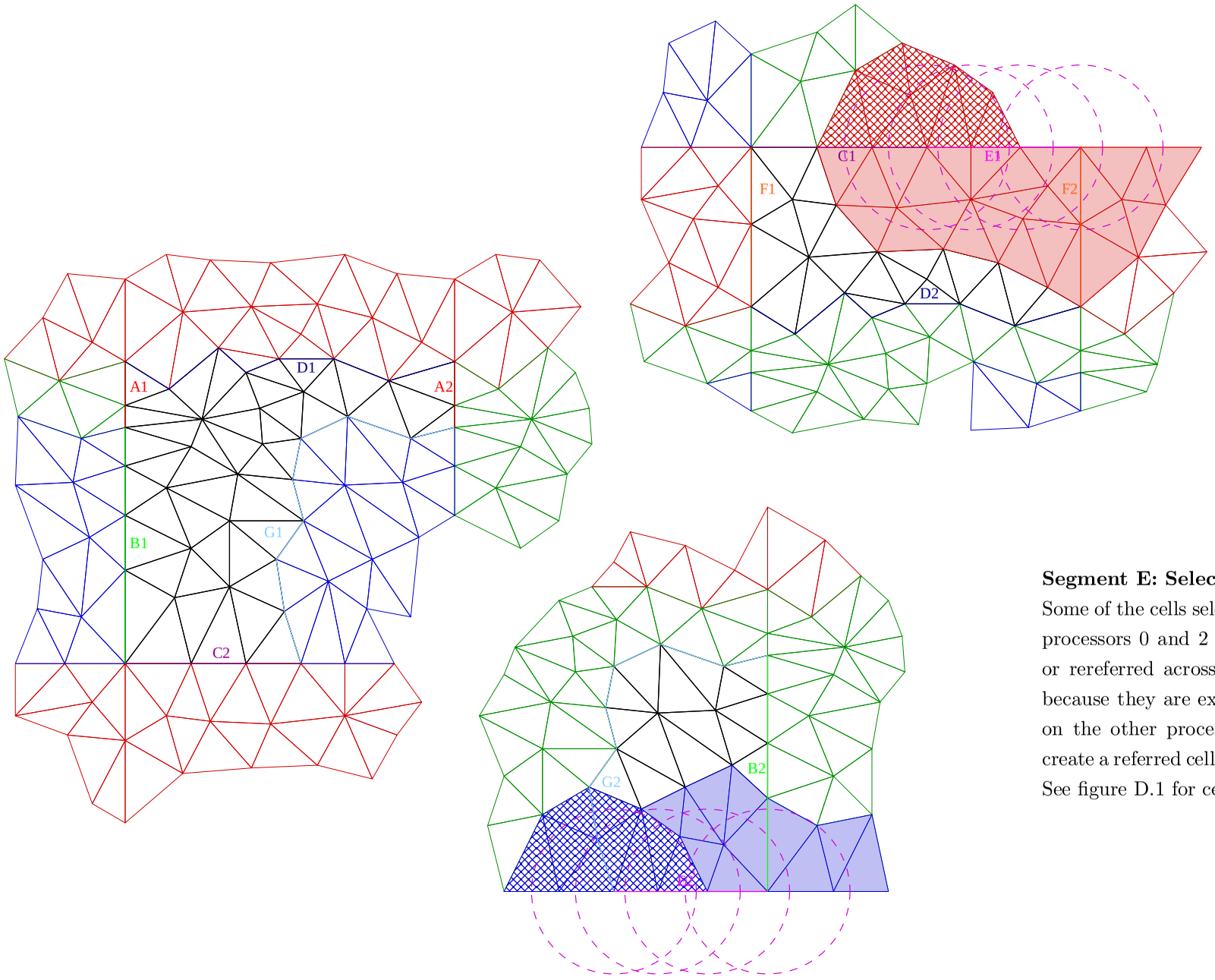
Segment G: Refer cells.



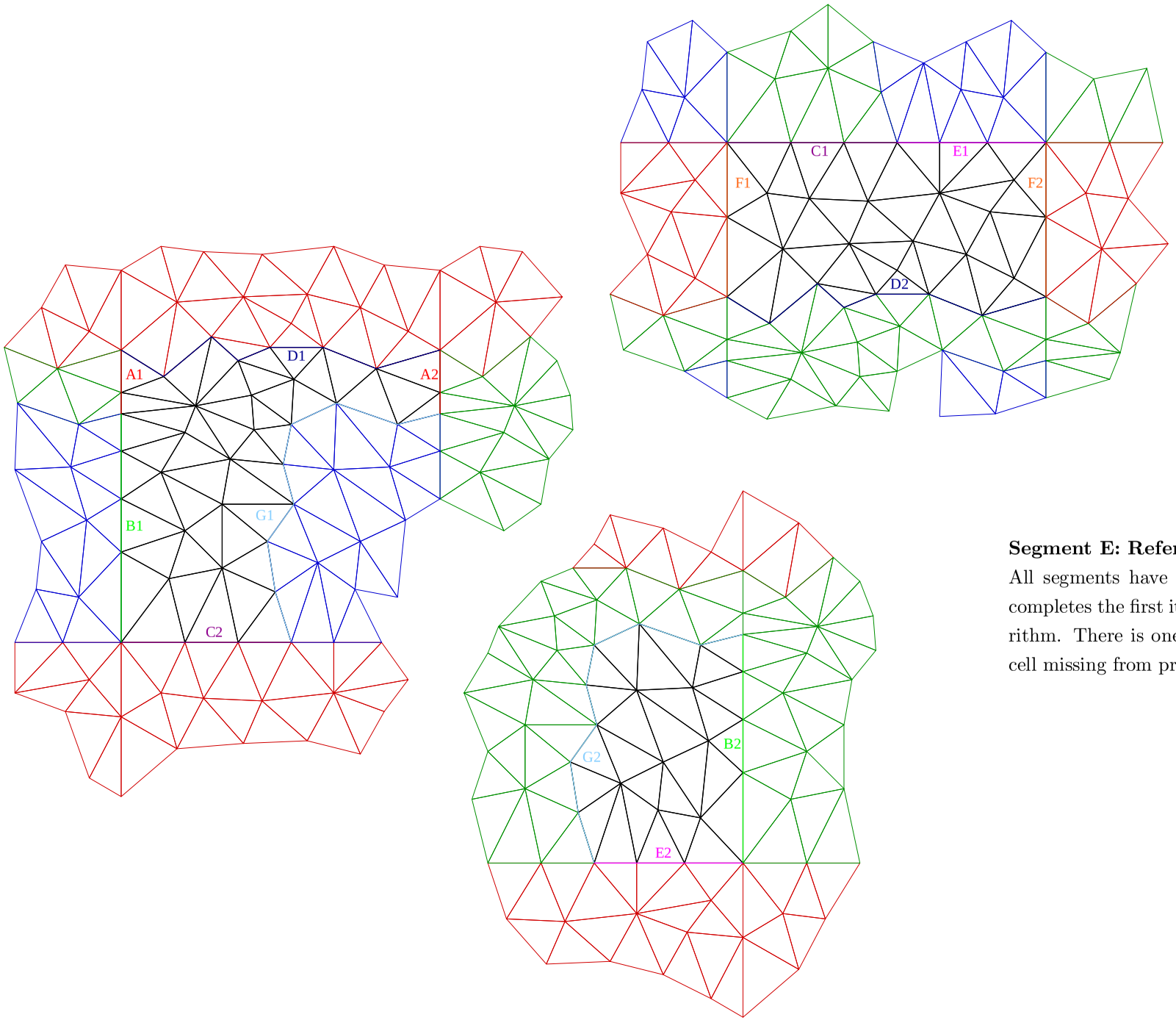
Segment C: Select cells.



Segment C: Refer cells.

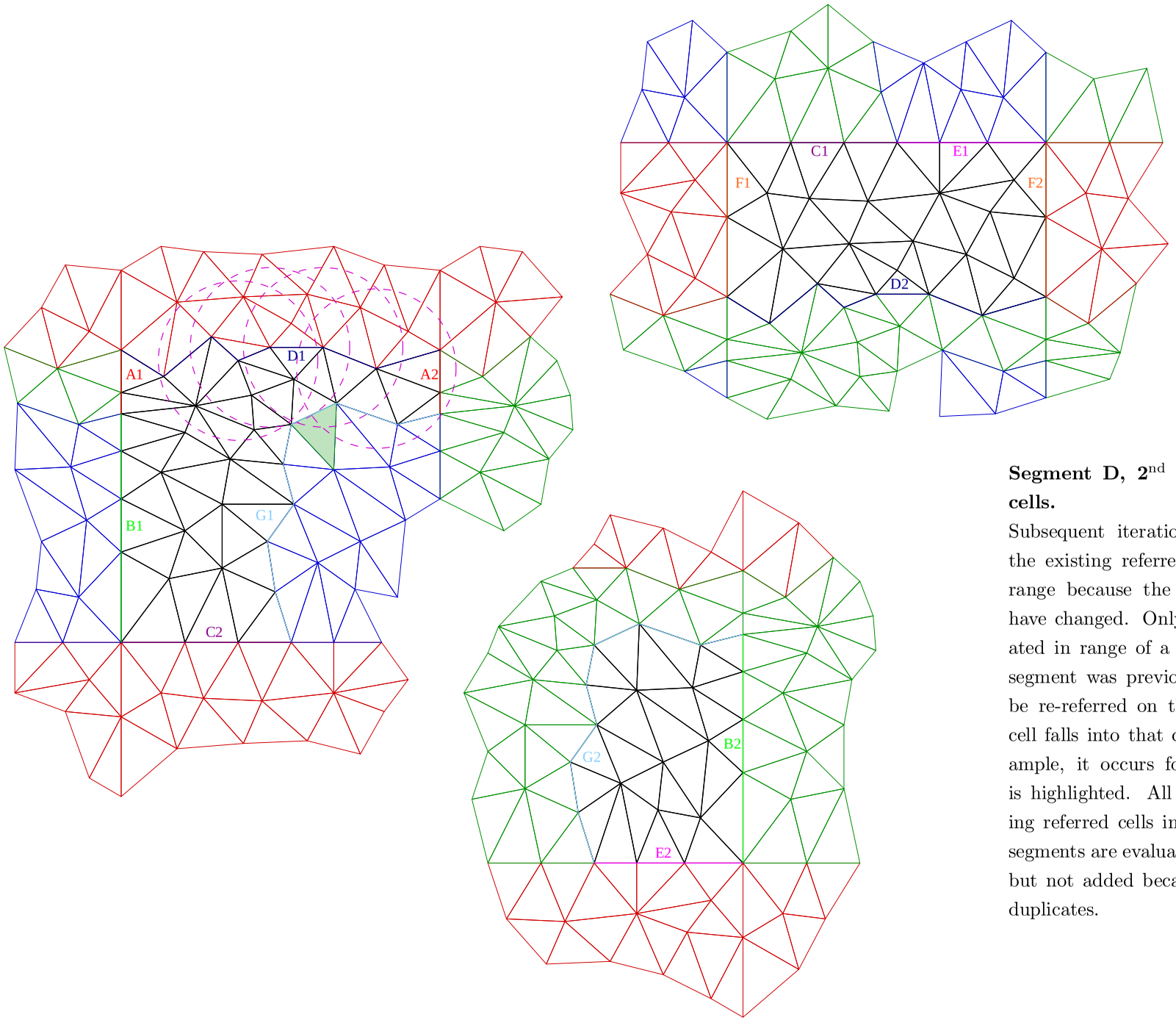


Segment E: Select cells.
Some of the cells selected for referral on processors 0 and 2 do not get referred or rereferred across segment E, either because they are existing referred cells on the other processor or they would create a referred cell on top of a real cell. See figure D.1 for cell colour coding.



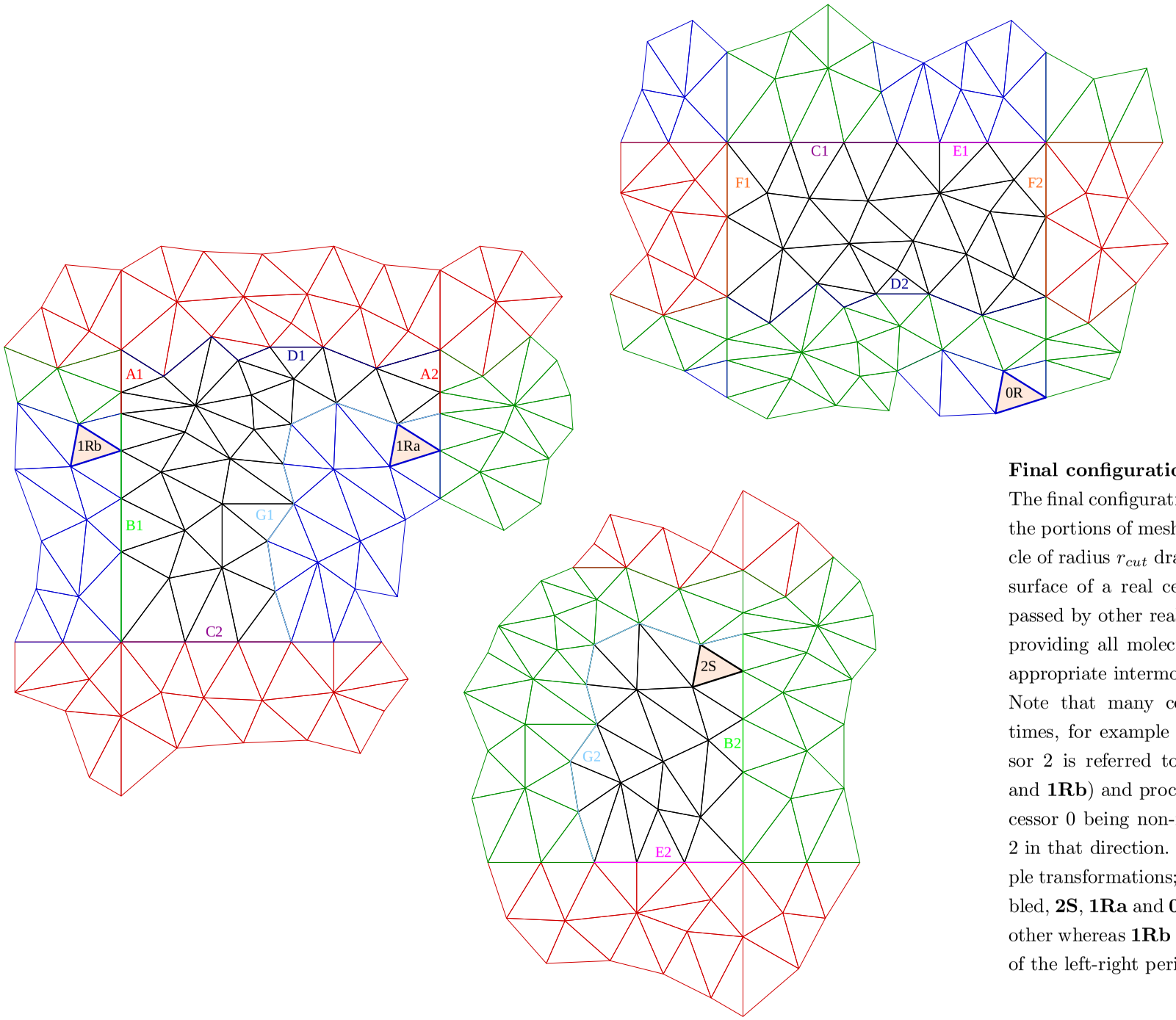
Segment E: Refer cells.

All segments have been assessed, this completes the first iteration of the algorithm. There is one necessary referred cell missing from processor 0.



Segment D, 2nd iteration: Select cells.

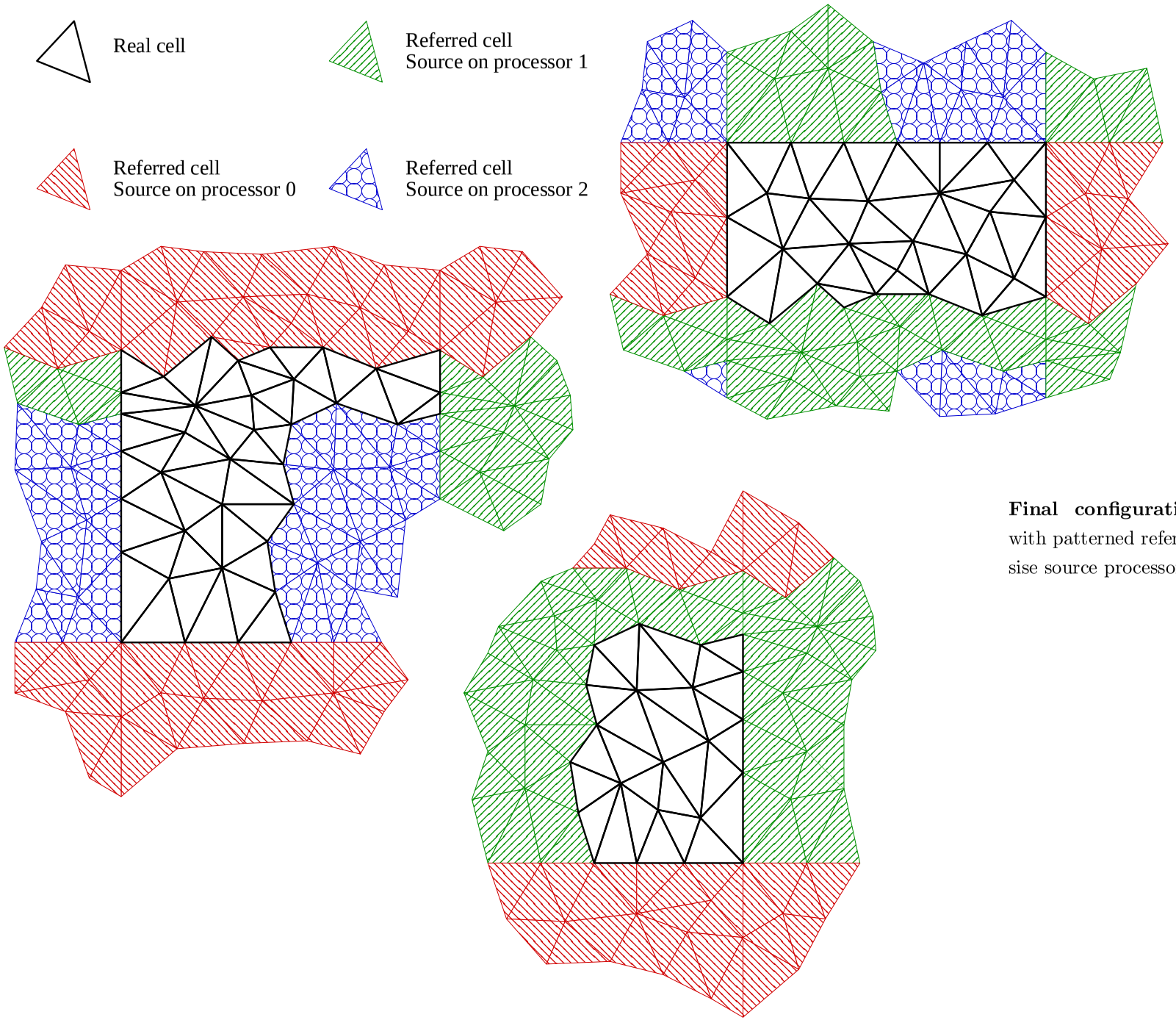
Subsequent iterations need only find the existing referred cells that are in range because the real cells will not have changed. Only referred cells created in range of a segment after that segment was previously evaluated will be re-referred on this iteration. One cell falls into that category in this example, it occurs for segment D, and is highlighted. All of the other existing referred cells in range of all other segments are evaluated in this iteration but not added because they would be duplicates.



Final configuration.

The final configuration of referred cells around the portions of mesh on each processor. A circle of radius r_{cut} drawn from any point on the surface of a real cell would be fully encompassed by other real or referred cells, thereby providing all molecules in that cell with the appropriate intermolecular interactions.

Note that many cells are referred multiple times, for example source cell **2S** on processor 2 is referred to processor 1 twice (**1Ra** and **1Rb**) and processor 0 (**0R**), despite processor 0 being non-neighbouring to processor 2 in that direction. The cells also have multiple transformations; if the mesh were reassembled, **2S**, **1Ra** and **0R** would lie on top of each other whereas **1Rb** would be on the other side of the left-right periodic boundary.



Final configuration. Re-coloured with patterned referred cells to emphasise source processor.

Appendix E

Cell referring transformations

A spatial transformation is required to refer a cell across a periodic or processor boundary. Figure E.1 shows the most general case of a cell being referred across a separated, non-parallel boundary, where,

$$
\begin{aligned}
\alpha^0 &= \text{cell with a face on one side of the boundary,}\\
\beta^0 &= \text{cell with a face on the other side of the boundary, coupled to } \alpha^0,\\
\alpha^1 &= \text{cell } \alpha^0 \text{ referred across the boundary,}\\
\mathbf{C} &= \text{face centre,}\\
\mathbf{n} &= \text{face normal unit vector,}\\
\mathbf{S}_{\alpha^0\beta^0} &= \mathbf{C}_{\beta^0} - \mathbf{C}_{\alpha^0}\text{, position shift from } \mathbf{C}_{\alpha^0} \text{ to } \mathbf{C}_{\beta^0},\\
\mathbf{R}_{\alpha^0\beta^0} &= \text{tensor required to rotate } -\mathbf{n}_{\alpha^0} \text{ to } \mathbf{n}_{\beta^0}\text{, given by,}
\end{aligned}
$$

$$
\mathbf{R}_{\alpha^0\beta^0} = -\left(\mathbf{n}_\alpha \cdot \mathbf{n}_\beta\right)\mathbf{I} + \left(1 + \mathbf{n}_\alpha \cdot \mathbf{n}_\beta\right)\left(\frac{\mathbf{n}_\alpha \times \mathbf{n}_\beta}{|\mathbf{n}_\alpha \times \mathbf{n}_\beta|}\right)^{\mathbf{2}} + \mathbf{n}_\alpha\mathbf{n}_\beta - \mathbf{n}_\beta\mathbf{n}_\alpha, \tag{E.1}
$$

where $\mathbf{I}$ is the identity tensor. The absolute position, $\mathbf{P}'$, of a molecule transformed across a boundary (with its containing cell) from its original position $\mathbf{P}$ is required. To derive the $\mathbf{P} \rightarrow \mathbf{P}'$ transform, first the position of $\mathbf{P}$ relative to the centre of the coupled face on α^0 is given by

$$
\mathbf{P} - \mathbf{C}_{\alpha^0}. \tag{E.2}
$$

This vector is rotated by the transformation tensor defined by the source and destination coupled face normals:

$$
\mathbf{R}_{\alpha^0\beta^0} \cdot (\mathbf{P} - \mathbf{C}_{\alpha^0}). \tag{E.3}
$$

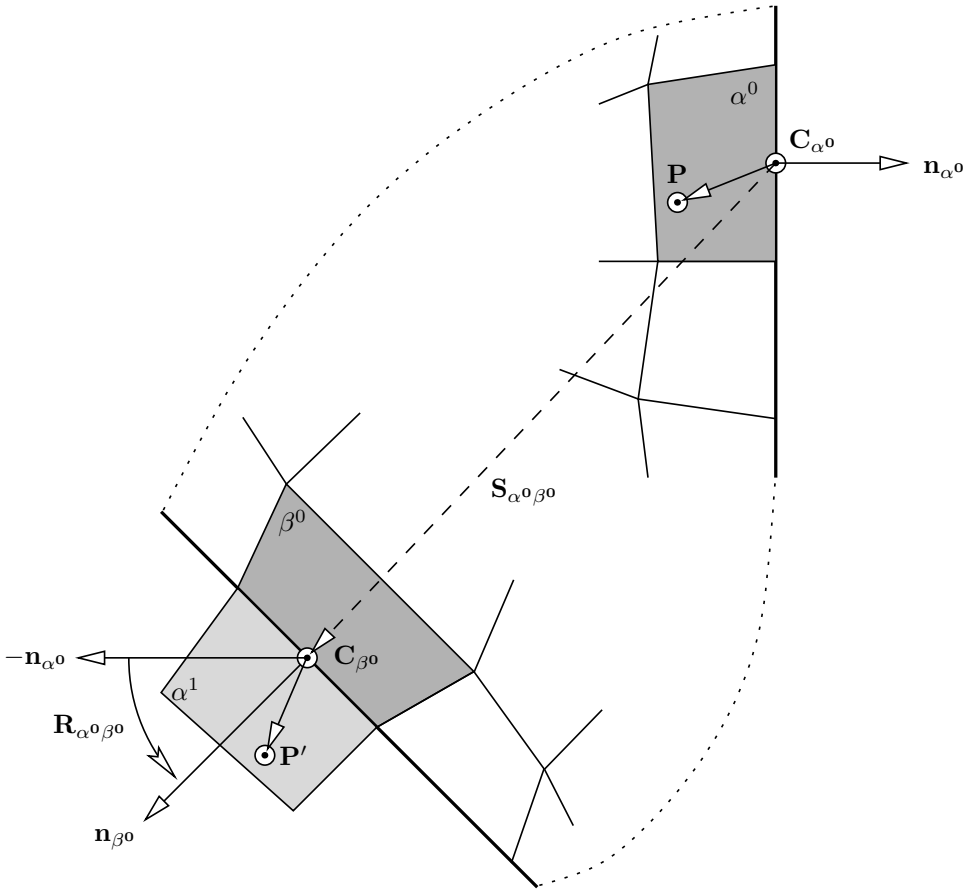


Figure E.1: A molecule at point $\mathbf{P}$ is referred across the boundary (heavy line) to $\mathbf{P}'$ by a transformation defined by the face centre/face normal vectors of the faces of cells α^0 and β^0 on the boundary. The mesh is rigid, so all points of cell α^1 have the same relative spatial relationship as α^0. The vertices of referred cell α^1 are calculated by the same process.

The rotated position is transformed back to global coordinates:

$$\mathbf{C}_{\alpha^0} + \mathbf{R}_{\alpha^0\beta^0} \cdot (\mathbf{P} - \mathbf{C}_{\alpha^0}), \tag{E.4}$$

and then shifted by the relative separation of the coupled face centres, i.e.

$$\begin{aligned}
\mathbf{P}' &= \mathbf{C}_{\alpha^0} + \mathbf{S}_{\alpha^0\beta^0} + \mathbf{R}_{\alpha^0\beta^0} \cdot (\mathbf{P} - \mathbf{C}_{\alpha^0}), \\
&= \mathbf{C}_{\alpha^0} + \mathbf{C}_{\beta^0} - \mathbf{C}_{\alpha^0} + \mathbf{R}_{\alpha^0\beta^0} \cdot (\mathbf{P} - \mathbf{C}_{\alpha^0}), \\
&= \mathbf{C}_{\beta^0} - \mathbf{R}_{\alpha^0\beta^0} \cdot \mathbf{C}_{\alpha^0} + \mathbf{R}_{\alpha^0\beta^0} \cdot \mathbf{P}.
\end{aligned} \tag{E.5}$$

The result is the same if the point is shifted by $\mathbf{S}_{\alpha^0\beta^0}$ prior to rotation by $\mathbf{R}_{\alpha^0\beta^0}$ around $\mathbf{C}_{\beta^0}$. The final position of $\mathbf{P}'$ is the same using any coupled face cen-

tre/face normal pair on the boundary. Therefore, all cells and all molecules referred across a particular boundary may use the transformation derived from one cell.

This result can be used to transform the positions of all of the molecules in a source cell to their correct position in a referred cell. Molecules being referred also operate on their own rotationally-dependent properties (e.g. angular orientation or velocity) using the rotation tensor. This transformation is suitable for multiple periodic boundaries and arbitrary mesh decompositions where existing referred cells must be re-referred by other boundaries. This creates the cell relationships across edges, corners, and also on non-neighbouring processors. See appendix D.4 for an example. Cell re-referring is achieved by writing equation (E.5) as a generic transform:

$$\mathbf{P}' = \mathbf{y} + \mathbf{R} \cdot \mathbf{P}, \tag{E.6}$$

where,

$$\mathbf{y} = \mathbf{C}_{\beta \mathfrak{o}} - \mathbf{R}_{\alpha \mathfrak{o} \beta \mathfrak{o}} \cdot \mathbf{C}_{\alpha \mathfrak{o}}, \tag{E.7}$$

$$\mathbf{R} = \mathbf{R}_{\alpha \mathfrak{o} \beta \mathfrak{o}}. \tag{E.8}$$

If $\mathbf{y}$ and $\mathbf{R}$ are the transformations required to move $\mathbf{P}$ to $\mathbf{P}'$, then transforming $\mathbf{P}'$ to $\mathbf{P}''$ using $\mathbf{y}'$ and $\mathbf{R}'$ gives:

$$\begin{aligned} \mathbf{P}'' &= \mathbf{y}' + \mathbf{R}' \cdot \mathbf{P}', \\ &= \mathbf{y}' + \mathbf{R}' \cdot \mathbf{y} + \mathbf{R}' \cdot \mathbf{R} \cdot \mathbf{P}, \\ &= \mathbf{y}'^{\star} + \mathbf{R}'^{\star} \cdot \mathbf{P}, \end{aligned} \tag{E.9}$$

reduced to the generic form by defining,

$$\mathbf{y}'^{\star} = \mathbf{y}' + \mathbf{R}' \cdot \mathbf{y}, \tag{E.10}$$

$$\mathbf{R}'^{\star} = \mathbf{R}' \cdot \mathbf{R}. \tag{E.11}$$

Further re-referrals can be reduced to a single transform in this way, with only a single vector/tensor pair required to be stored by the referred cell at any stage. Any number of cell referring transformations may be made sequentially, but the resulting transformation takes the initial source position ($\mathbf{P}$) and directly transforms it to the final destination regardless of, and without having to intermedi-

ately visit, the other referred locations along the way.

Appendix F

Non-equilibrium gas dynamics

There is an opportunity to use MD to explore some important and unresolved issues in rarefied gas dynamics, especially in micro and nano systems [42]. Rarefied conditions in a gas are characterised by high Knudsen number, $Kn = \lambda/L$, which can arise from a large mean free path, λ, (low pressure gas, e.g. high altitude aeronautics) or small (micro or nano) characteristic length, L.

Many of the unusual effects seen at surfaces in microsystems are thought to arise from the gas not being in equilibrium. To test this, fundamental properties of a gas — the molecular velocity distribution and the distribution of free path between collisions — and their spatial variation, can be measured from MD simulations of a moving gas in the presence of solid walls. This information cannot be obtained except by direct simulation because the velocity distribution of the gas, and many of its dependent parameters cannot be measured experimentally. Only macroscopic properties can be measured reliably, such as average mass flowrate through a channel. Interpretation of these results, however, is problematic because there can be multiple causes of deviation of these quantities from those expected from macroscopic theory, separating out their relative contribution is difficult.

Previously reported simulations of the distribution of free path between collisions [130] were carried out with the interaction between molecules represented by step intermolecular potentials, such as the hard sphere or square well models. The instance of a 'collision' is unambiguous with a step potential, but continuous potentials have been used here. A collision is defined to have occurred at the position and time when molecules first move closer than the cut-off radius of their intermolecular pair potential, and the free path and free time between collisions

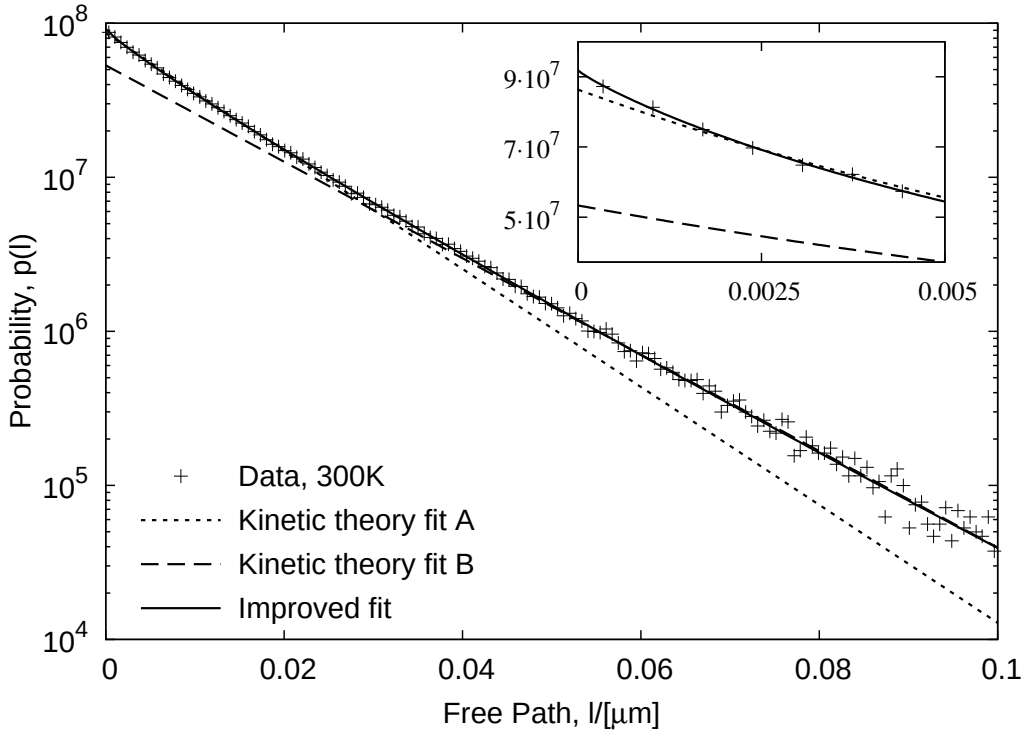


Figure F.1: Free path distribution — LJ gas, 300K. Kinetic theory predicts an exponential relationship (equation (F.1)), however we observe that this cannot represent the very short path (inset), short path (A) and long path (B) characteristics simultaneously. An improved fit is found using equation (F.2).

is calculated as being the interval between these interaction start points.

Figure F.1 shows the free path distribution for a simulation of 3375 Lennard-Jones atoms ($r_{cut} = 0.85$nm) at a density of 1.7kg/m^3, equilibrated to 300K in a periodic domain, resulting in a pressure of 1.05bar — approximately atmospheric conditions. The results are averaged over 400000 timesteps of 10.8fs each. Kinetic theory [182] predicts that the free path distribution, $p(l)$ is of the form

$$p(l) = a_0 e^{-\frac{l}{\lambda_0}}, \tag{F.1}$$

where a_0 is chosen to normalise the distribution and λ_0 is the mean free path. This single exponential functional form does not fit the data particularly well across the whole free path range. Modifying the functional form to

$$p(l) = a_0 e^{-\frac{l}{\lambda_0}} + a_1 e^{-\frac{l^r}{\lambda_1}}, \tag{F.2}$$

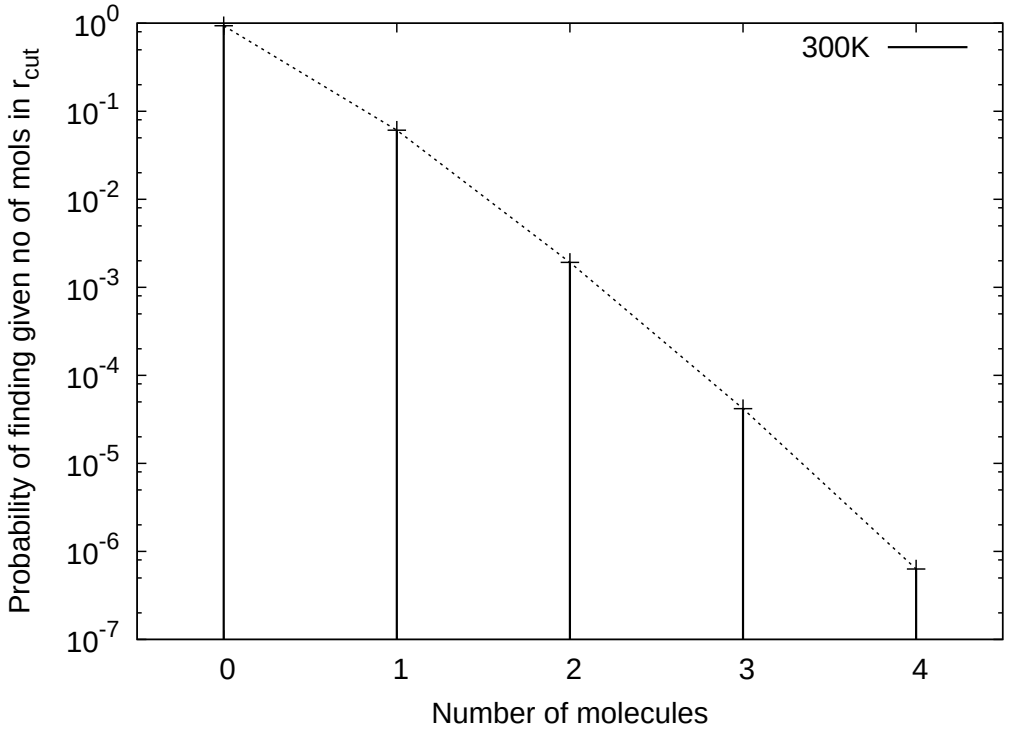


Figure F.2: Probability of finding a given number of other molecules simultaneously in a molecule's intermolecular potential — LJ gas, 300K.

gives a significantly better fit of the data, although no physical significance has been ascribed to the additional parameters. It is possible that the definition of a collision employed here is responsible for the deviation from exponential form. It should be noted that the free path result is dependent on the (arbitrary) cut–off radius employed. This is a recognised problem when using continuous classical intermolecular potentials [86]. However, what is most important is the difference between free path distribution in a bulk fluid and near a surface, so as long as a consistent cut–off radius and collision definition is used, this should not be an issue.

The collision handling algorithm implemented allows the evaluation of the time spent and distance travelled in collision for an arbitrary number of simultaneous collisions — functioning correctly at liquid densities, where any molecule is typically in simultaneous 'collision' with approximately 50 others. Figure F.2 shows the probability of finding a given number of other molecules in a molecule's force field. This demonstrates that, when using a realistic intermolecular potential (including an attractive portion) instead of simple repulsive model, then there is a

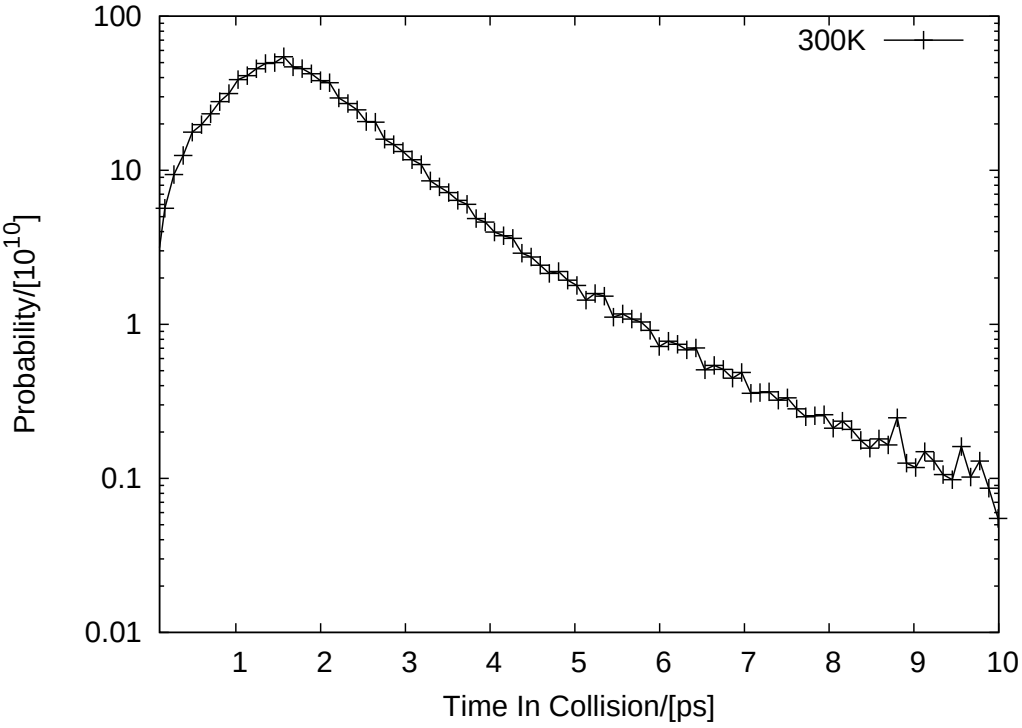


Figure F.3: Distribution of time spent in collision — LJ gas, 300K.

significant chance of observing collisions that involve more than two molecules (a binary collision). These results show that 3.1% of all collision events will involve three or more molecules. This has direct implications for identifying the limit of the dilute gas approximation employed when deriving the Boltzmann equation.

The probability function for the time that molecules spend in each other's intermolecular potentials is shown on figure F.3. The data has a very long 'tail'; the probability of spending a long period in collision is relatively high (pairs of molecules were observed remaining in collision for longer than 450ps). This is interpreted as molecules forming temporary pairings, via the attractive part of their potentials. Another of the assumptions underpinning the Boltzmann equation is that of molecular chaos, where the motion of individual particles is uncorrelated, something which is not applicable to molecules which spend significant periods interacting.

Spatially resolved data for microscopic state of a gas near to a surface will provide insight into two unresolved issues in practical micro system design:

1. slip and thermal transpiration;

2. the Knudsen layer.

Under high Kn conditions, the velocity of a gas is not the same as an adjacent surface — it can 'slip' over it. This process is also dependent on the local temperature gradient of the surface, because the gas has a tendency to flow from cold to hot, an effect known as thermal transpiration. Maxwell [43] produced a velocity boundary condition which can successfully describe slip and thermal transpiration flows [183]. This relies, however, on the inclusion in the slip equations of a tangential momentum accommodation coefficient (TMAC) — which is an adjustable phenomenological parameter depending on many details of the wall geometry and composition. TMAC values cannot be easily measured by experiment or theoretically deduced. They are usually assumed to be a constant for the purposes of analysis. This may be masking the effects of the non–equilibrium nature of gas–solid interactions. It was acknowledged by Maxwell [43] that,

> "... it is almost certain that the stratum of gas nearest to a solid body is in a very different condition from the rest of the gas."

MD can directly probe the mechanisms underlying slip and thermal transpiration. The main strength of MD is that it is not confined to simple (monoatomic) gases or simple 'equivalent' models for surfaces. The effects of surface roughness; velocity gradient, wall model; gas type — diatomic gases or a mixture representing air; temperature and temperature gradient, for example, can all be studied.

Within approximately one mean free path of a surface, the linear relationship between shear stress and velocity gradient breaks down — this region is known as the Knudsen layer. It is possible to model the effect of the Knudsen layer by phenomenologically scaling the local constitutive relations of the gas — creating a non–Newtonian fluid with a position dependent viscosity [184]. This has been shown to be an effective technique, although more data gathered by MD would extend the applicability of the technique to complex geometries and more gas species.

Appendix G

Introduction to OpenFOAM

G.1 Overview

OpenFOAM is an open source[1] code intended to perform large scale engineering continuum computational fluid mechanics (CFD) simulations. It has been applied to industrial problems such as automotive external aerodynamics and internal combustion engine flows.

OpenFOAM comprises a collection of C++ libraries and executables. Executables are built from the libraries and are divided into solvers and utilities. Solvers simulate a particular physical problem and utilities manipulate the geometry of a simulation case, or the results of a solver, to perform pre-processing and post-processing operations. Solvers for a wide range of continuum mechanics problems are included in the code: incompressible and compressible fluid flows including turbulence and non-Newtonian fluids; multiphase flows; direct numerical simulation (DNS) and large eddy simulation (LES); combustion; heat transfer; electromagnetics; solid dynamics and financial modelling. There is also a suite of tools for creating and manipulating mesh geometries as well as converting them to and from external formats.

Users may create new solvers and utilities using only abstracted, top-level syntax without knowledge of the implementation details of built-in algorithms and operations. For example, solvers are created by typing continuum mechanics equations in tensor form into the solver code. These are discretised and solved, in parallel if necessary, by the underlying classes, unseen by the user.

For further information, and to obtain a copy of OpenFOAM and its docu-

[1]Released under the GNU General Public License, www.gnu.org/licenses.

mentation, see www.openfoam.org.

G.2 Modular code structure

OpenFOAM has a modular and extensible code structure which makes it amenable to performing a wider range of physical simulations than its core continuum mechanics remit. There are many C++ classes that deal with file IO and establishing controls for run-length, timestep, the interval for writing to disk, for example, which are common to every type of simulation and may be reused.

Similarly, every simulation requires the geometry that is being simulated to be defined. The unstructured arbitrary polyhedral mesh description in OpenFOAM is powerful and flexible, providing functions for extracting geometrical and connectivity information. For example: what are the face centre positions for the faces of particular cell? Which cells are connected to a particular vertex? See section 6.1 of the OpenFOAM User Guide and section 2.3 of the OpenFOAM Programmers Guide [5] for details of the mesh description.

OpenFOAM uses geometrical decomposition to implement distributed memory parallelisation, and every component is coded at its most basic level to operate in parallel. A good example is the Lagrangian particle tracking system that the molecular dynamics code described has been based on. The basic particle class represents an abstract, generic discrete particle and implements the functionality to track it through the mesh, identify when it hits boundaries, and transfer it between processors when it crosses between the portions of a spatially decomposed mesh. All of this functionality is templated, so that when a specific type of particle is derived from the basic particle it requires only the additional data and modelling pertinent to it. A molecule, for example, requires velocity, acceleration and species identity information and the functionality to calculate intermolecular pair forces. The molecule class inherits the basic motion and parallel functionality without any changes being made to the basic particle class.

All of this functionality may be implemented independently of the continuum mechanics solvers. All of the components link together easily however, so, for example, it is straightforward to impose the effect of a continuum field on a discrete particle and *vice versa* without requiring changes to the underlying libraries for either component.

G.3 Why C++?

Creating a flexible, comprehensible and expandable top-level syntax to allow rapid user-creation of interoperable libraries and executables, without requiring intimate knowledge of, and requiring changes to be made to the core code, requires many features of the C++ programming language. These primarily are object orientation, encapsulation and code reuse, polymorphism and generic programming by inheritance and templated classes. For a more detailed discussion of the choice of language see section 3.1 of the OpenFOAM User Guide [5].

It is worth noting that, contrary to traditional impressions, particularly for x86 architecture computers, C++, together with a good compiler does not produce slow software, and FORTRAN does not represent the absolute standard for performance [185]. Properly written and compiled C++ executables are fast.