2020-7

Red Blood Cell Dynamics on Non-Uniform Grids using a Lattice Boltzmann Flux Solver and a Spring-Particle Red Blood Cell Model

Brendan Walshe

Red Blood Cell Dynamics on Non-Uniform Grids using a Lattice Boltzmann Flux Solver and a Spring-Particle Red Blood Cell Model

by

Brendan Walsh

Supervisor: Prof. Fergal J. Boyle

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

School of Mechanical & Design Engineering
College of Engineering and Built Environment
and
Environmental Social & Health Institute

July 2020

Abstract

The Computational Haemodynamics Research Group (CHRG) in Technological University Dublin is developing a computational fluid dynamics (CFD) software package aimed specifically at physiologically-realistic modelling of blood flow. A physiologically-realistic model of blood flow involves calculating the deformation of individual red blood cells (RBCs) and the contribution of this deformation to the overall blood flow. The CHRG has developed an enhanced spring-particle RBC structural model that is capable of modelling the full stomatocyte-discocyte-echinocyte (SDE) transformation. This RBC model, incorporated into a fluid dynamics solver, will provide a physiologically-realistic blood flow model. In this work the overall plasma flow is modelled using a novel technique: the lattice Boltzmann flux solver (LBFS). This is an innovative approach to solving the Navier-Stokes (N-S) equations for fluid flow. It involves solving the macroscopic equations using the finite volume method (FVM) and calculating the flux across the cell interfaces via a local reconstruction of the lattice Boltzmann equation (LBE). Fluid-struture interaction between the RBC and the plasma is captured by coupling the RBC solver to the LBFS via the immersed boundary method (IBM). Numerical experiments investigating RBC dynamics are performed using non-uniform grids and validated against existing experimental data in the literature. Finally all numerical solvers are developed using general purpose GPU programming (GPGPU) and this is shown to accelerate simulation runtimes significantly.

Declaration of Authorship

I certify that this thesis which I now submit for examination for the award of doctor of philosophy, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for graduate study by research of Technological University Dublin and has not been submitted in whole or in part for another award in any other third level institution.

The work reported on in this thesis conforms to the principles and requirements of the TU Dublin's guidelines for ethics in research.

TU Dublin has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Signature of Candidate: _____

Date: _____

Acknowledgements

I would foremost like to thank my supervisor Prof. Fergal Boyle for his guidance, input and support over the last few years. He has always made himself available and his rigorous feedback is very much appreciated.

My colleagues in the CHRG have been great sounding boards and I'd like to thank Ger, Osay, Joe and Ming Zhu for all the support they've given over the years.

I'd like to thank TU Dublin for funding this research through the Fiosraigh scholarship scheme. I'd also like to thank ESHI for providing the great facilities within which to perform my research and also the support staff; Jesus, Kevin, Claudio and Ola for ensuring ESHI meets a modern researchers needs. I'd like to thank NVIDIA for the kind donation of a Tesla K40c which was used during this work. I'd also like to acknowledge ICHEC for granting access to the Irish supercomputer Kay and the excellent support service offered with the initial onboarding process. I'd like to thank Prof. David Kennedy for giving me the opportunity to lecture in parallel to my research. I'd also like to thank Dr. Yan Wang for his generous time in responding to my queries on the nuances of the Lattice Boltzmann Flux Solver.

Finally I'd like to thank my friends and family for the good times and support over the last few years. I look forward to the future when conversations won't start with "When will you be finished?" or "What will you do when your done?".

Table of Contents

List of Figures

List of Tables

Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
ACM	Artificial Compressibility Methods
ADE-SP	Area-Difference-Energy Spring-Particle
BEM	Boundary Element Methods
BGK	Bhatnagar-Gross-Krook
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Levy
CG-SP	Coarse Grained Spring-Particle
CHRG	Computational Haemodynamics Research Group
CPU	Central Processing Units
DF-IBM	Direct-Forcing-IBM
DPD	Dissipative Particle Dynamics
FDM	Finite Difference Method
FDM	Fictitious Domain Method
FEM	Finite Element Methods
FSI	Fluid-Structure-Interaction
FVLBM	Finite Volume - Lattice Boltzmann Methods
FVM	Finite Volume Method
GLSQ	Green-Gauss/Weighted-Least-Squares
GPGPU	General Purpose GPU Programming
GPU	Graphics Processing Units
HPC	High Performance Computing
IBM	Immersed Boundary Method
IFEM	Immersed Finite Element Method
LBE	Lattice Boltzmann Equation
LBFS	Lattice Boltzmann Flux Solver

LBM	Lattice Boltzmann Method
LBS	Lattice Boltzmann Solution
LVE	Low Viscosity Ektacytometry
MAD	Membrane Area Difference
MIC	Many Integrated Core
MPCD	Multi-Particle Collision Dynamics
MPSI	Moving Particle Semi-Implicit
N-S	Navier-Stokes
PBS	Phosphate-Buffered-Saline
PLBFS	Preconditioned Lattice Boltzmann Flux Solver
PM	Plasma Membrane
RBC	Red Blood Cell
RK4	Fourth-Order Runge-Kutta
RMS	Root Mean Square
RPPM	Reproducing Polynomial Particle Method
SDE	Stomatocyte-Discocyte-Echinocyte
SPH	Smoothed Particle Hydrodynamics
TFLOPS	Terra Floating Point Operations Per Second
VI-FS	Velocity Interpolation and Force Spreading
WLC	Worm Like Chains

Symbols

Scalers

T_K	Absolute temperature
k_B	Boltzmann constant
x, y, z	Cartesian coordinates
S	Cell surface
V	Cell volume
u, v, w	Components of the velocity vector $\mathbf{v}$ in the x, y and z directions respectively
CN	Condition number
ρ	Density
f_α	Density distribution function in the α direction
μ	Dynamic viscosity
λ	Eigenvalues of flux jacobian
f_α^{eq}	Equilibrium density distribution function in the α direction
P_ω	Fourier form of the numerical amplification operator
K_A	Global area modulus
$K_{B,ADE}$	Global bending modulus
E_S	Helmholtz energy due to the in-plane shearing resistance of the cytoskeleton
E_B	Helmholtz energy due to out-of-plane bending resistance in the pm
E_A	Helmholtz energy due to the conservation of area constraint

E_V	Helmholtz energy due to the conservation of volume constraint
E_{RBC}	Helmholtz energy of the instantaneous shape of the RBC model
A_k	Instantaneous area of triangle k
C_j	Instantaneous curvature of the particle j
L_s	Instantaneous length of the spring s
ΔA_m	Instantaneous MAD of the RBC
A_{RBC}	Instantaneous total area of the RBC
V_{RBC}	Instantaneous volume of the RBC
$\mathcal{L}_I$	Inviscid flux spatial discretisation operator
ν	Kinematic viscosity
δx	Lattice spacing along each cartesian axis
c	Lattice velocity
w_α	Lattice velocity weighting
$K_{A,k}$	Local area modulus
K_B	Local bending modulus
Ma	Mach number
f_α^{neq}	Non-equilibrium density distribution function in the α direction
N_{faces}	Number of cell faces on a cell
N	Number of velocities in the lattice model
g	Numerical amplification factor
L_P	Persistent length of the wlc springs
$\overline{p}$	Preconditioned pressure
τ_p	Preconditioned relaxation factor
γ	Preconditioning parameter
λ_s	Ratio of the instantaneous length of the spring to the contour length of the spring
Γ_{vc}	Ratio of viscous spectral radius to convective spectral radius
H_m	RBC membrane thickness
$A_{k,0}$	Reference area of triangle k
$C_{j,0}$	Reference curvature of the particle j
D	Reference length
$\Delta A_{m,0}$	Reference MAD of the RBC

$A_{RBC,0}$	Reference total area of the RBC
$V_{RBC,0}$	Reference volume of the RBC
Re	Reynolds number
K_S	Shear modulus.
Λ_c	Spectral radius of convective flux jacobian
Λ_v	Spectral radius of inviscid flux jacobian
c_s	Speed of sound in the lattice grid
τ_s	Standard relaxation factor
p	Static pressure
ψ	Streamfunction
δt	Streaming time step
$L_{c,s}$	The contour length of the spring s
t	Time
Δt	Time integration time step
λ_μ	Viscosity ratio between cytosol and external fluid
$\mathcal{L}_V$	Viscous flux spatial discretisation operator
$\mathbf{F}_{V,js}$	Viscous force at particle j due to spring s
K_V	Volume constraint modulus
Wo	Womersley number
$e_{\alpha,x}$, $e_{\alpha,y}$ and $e_{\alpha,z}$	x, y and z components respectively of the particle velocity vector $\mathbf{e}_\alpha$

Tensors

$\mathbf{S}$	Cell Surface
$\mathbf{m}_k$	Centroid of triangle k
$\mathbf{F}_{A,j}$	Conservative force due to the area constraint at particle j
$\mathbf{F}_{V,j}$	Conservative force due to the volume constraint at particle j
$\mathbf{r}$	Eulerian coordinates
$\mathbf{F}$	Flux Tensor
$\mathbf{F}_a$	Force Density per unit Area
$\mathbf{F}_{vol}$	Force Density per unit Volume
$\mathbf{F}_I$	Inviscid Flux Tensor

$\mathbf{R}$	Lagrangian coordinates
$\boldsymbol{\xi}_k$	Normal to triangle k
$\mathbf{n}$	Outward normal to element of area ds
$\mathbf{e}_\alpha$	Particle velocity vector in *alpha* direction
$\mathbf{R(Q)}$	Residual Function
$\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$	Unit vectors in the x, y and z directions
$\mathbf{Q}$	Vector of conserved variables $[\rho, \rho u, \rho v, \rho w]$
$\mathbf{V}$	Velocity vector
$\mathbf{F}_V$	Viscous Flux Tensor
$\boldsymbol{\zeta}$	Vorticity Vector

Subscripts

i	cell index
i, n	n^{th} face on cell i
$*$	Non-Dimensional Variable
j	Lagrangian node index
h	Size of the stencil of the Dirac delta distribution
k	Triangle index
s	Spring index

Other

$\mathbf{P}$	Preconditioning matrix
$g_{j,h}$	Set of Eulerian cells that lie within the stencil of the Dirac delta distribution
$G_{\mathbf{r},h}$	Set of Lagrangian nodes that lie in the stencil of a cell $\mathbf{r}$

Chapter 1

Introduction and Literature Review

1.1 Introduction

In this chapter, the underlying research motivation of this thesis is introduced along with the different areas of the research problem. These areas include the rheology and haemodynamics of blood flow, the structure of red blood cells (RBCs), the various existing approaches to structural modelling of RBCs, the various existing approaches to the two-phase modelling of blood flow, the various existing approaches to modelling fluid flow and runtime acceleration of numerical solvers. The aims of the research are then presented and the original contributions of the research are then discussed. The structure of the thesis is described and the publications derived from the research are noted.

1.2 Research Motivation

The circulatory system is a key organ system that allows blood to circulate and transport oxygen, nutrients, waste products, molecules and cells critical to the function of an organism. Cardiovascular diseases are the biggest cause of death in the world [13]. As a result there is significant interest and research into understanding the circulatory system and developing more sophisticated approaches to combating cardiovascular diseases. A key element of this research is creating a full

understanding of blood flow that can then be used in real world applications such as biomedical device design and drug delivery. Empirical indices, *in-vitro* and *in-vivo* experimental tests have all been developed to create a better understanding of blood flow to aid real world biomedical design applications [14]. However these approaches all have deficiencies, empirical indices lack generality outside of laboratory conditions and *in-vitro* and *in-vivo* methods are very expensive and time consuming especially in the context of the multiple design iterations involved in modern day engineering design processes. As a result there is a demand to develop accurate *in-silico* models of blood flow which would enable bespoke modelling of real world problems and reduce the need for costly and time-consuming *in-vivo* and *in-vitro* models.

Computational fluid dynamics (CFD) simulations are increasingly being used to model blood flow in real world applications such as deployed stents and heart valves. CFD packages such as ANSYS CFX [15], OpenFoam [16] [17] and Star-CD [18] have all been used to model macro-scale problems. However these packages use the assumption that blood is a non-Newtonian, incompressible and homogenous fluid. Blood is far from homogeneous in nature with 37-54% of its volume consisting of suspended RBCs in plasma [19]. This assumption of homogeneity leads to very questionable conclusions when existing commercial packages are used to model the physiological behaviour of blood. This assumption ignores that RBCs are deformable objects that are free to move through the plasma. RBC location and velocities can have a key role in the development of certain cardiovascular diseases e.g. thrombosis [20]. However tracking and calculating the deformation of individual RBCs is an extremely computationally expensive process. The development of a numerical blood flow model which captures the deformations of individual RBCs with the computational efficiency to model real world problems would be a major advancement to combatting cardiovascular diseases.

1.3 Rheology and Haemodynamics of Blood Flow

As mentioned above, blood is a suspension of cellular elements in plasma. These cellular elements include RBCs (erythrocytes), white blood cells (leukocytes) and platelets. RBCs account for approximately 99.9% of blood cells [19]. Plasma is the suspending phase of the cellular elements and can be considered to behave as a Newtonian fluid [21] and accounts for roughly 45-63% of the volume of blood.

The viscosity of plasma changes with temperature but is in the range of 1.16 - 1.35 mPa s at 37 °C [22]. However the viscosity of blood behaves in a non-Newtonian fashion. Experimental data from rotational viscometers shows blood having a high apparent viscosity at low shear rates and low apparent viscosity at high shear rates [23] [24]. At high shear rates of 100 - 200 s^{-1}, the viscosity approaches a minimum of 4-5 mPa s at 37 °C. The viscosity of blood is also shown to be highly dependent on hematocrit with increasing amounts of RBCs increasingly disturbing the flow lines [25]. At medium to high shear rates this effect is more pronounced with a unit increase in hematocrit (the ratio of the volume of red blood cells to the total volume of blood) ,e.g 43 to 44%, increasing blood viscosity by 4% [21]. The rheological properties of the RBCs also influences the viscosity of blood. RBCs are highly deformable bodies and exhibit different behaviours at varying shear rates. At high shear rates, RBCs deform and tank tread (RBC membrane revolves around RBC centroid with constant orientation) in shear flow. This has the effect of producing a comparatively low viscosity for blood flow at high shear rates when compared to other suspensions of similar concentrations [26]. Aggregation is another influencing factor on blood viscosity. At lower shear rates, lower levels of deformation take place and at levels under 50 s^{-1} RBCs tend to aggregate into linear stacks called rouleaux. The effect of the stacks is to create a higher drag surface than individual isolated RBCS, which has the effect of increasing the viscosity of blood [26]. The literature shows that the viscosity of blood is highly dependent on the hematocrit and rheological properties of RBCs such as deformability and aggregation. This supports the need for a numerical blood flow solver which captures the deformation and tracking of individual RBCs.

1.4 Red Blood Cell Structure

A normal RBC is a nucleus free cell that is biconcave in shape (see Figure 1.1). It has diameter of approximately 8.0 µm and a thickness of approximately 2.0 µm [27]. Structurally it is a membrane bounded capsule with a liquid core. This liquid core consists of cytosol which is a mixture of haemoglobin and enzymes and is considered to be a Newtonian fluid [28]. The cytosol has a viscosity that is approximately five times larger than that of plasma [29]. This low viscosity ratio of haemoglobin to plasma enables a highly responsive deformation of the RBC in response to external loading. The membrane consists of an outer layer of

plasma membrane (PM) and an inner layer of cytoskeleton (see Figure 1.1). Each of these structural features have distinct properties that need to be captured when predicting the biomechanics of a RBC.

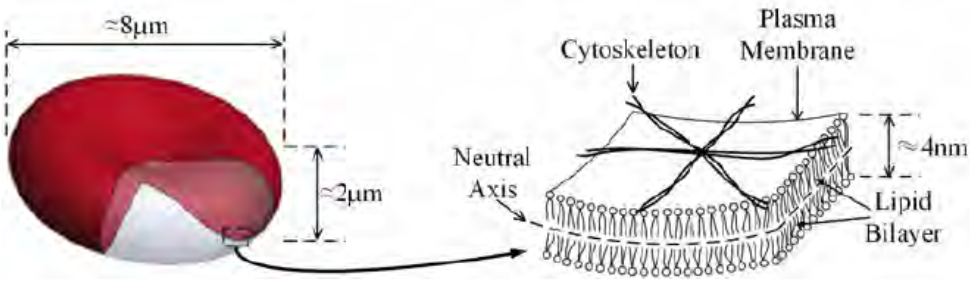


FIGURE 1.1: Illustrations of a healthy RBC and the RBC membrane. The main structural feature of the PM is the lipid bilayer, while the cytoskeleton is a hyper-elastic-behaving network attached to the PM via anchoring proteins. [1]

A more detailed schematic of the PM can be seen in Figure 1.2. The PM acts as the main barrier between the extracellular and intracellular fluids. The fluid mosaic model is the most commonly accepted approach to describe the structure of the PM [30]. This model states that the PM is a mosaic comprised of phospholipids, proteins, cholesterol and carbohydrates. These components are all free to move around each other and exhibit great fluidity [31]. The phospholipids and cholesterol are the main constituent of the lipid bilayer. The bilayer forms the basic structure of the membrane and acts as semi-permeable barrier to the extracellular environment. Dispersed within the bilayer are then transmembrane proteins and carbohydrates. The formation of the bilayer is due to the amphipathic nature of the lipids. The lipid heads are hydrophilic while the lipid tails are hydrophobic, resulting in the lipid bilayer structure. Of structural significance is that constituents of the PM, while tightly attached, lack physical inter-constituent connections. Hence the PM lacks a constraint against in-plane deformation due to loading and the PM exhibits surface-area incompressibility [32]. Also of structural significance is the fact that the lipid bilayer exhibits high compression resistance in thickness [33]. The consequence of this is that the removal, insertion or exchange of the constituents of the bilayer results in volume changes of the lipid bilayer and changes in membrane area difference (MAD), *i.e* area difference between the inner and outer layers of the bilayer. In general constituents of both layers of the lipid bilayer are considered to be approximately the same. Therefore the MAD of the bilayer is preferable to be zero. The cytoskeleton is another layer

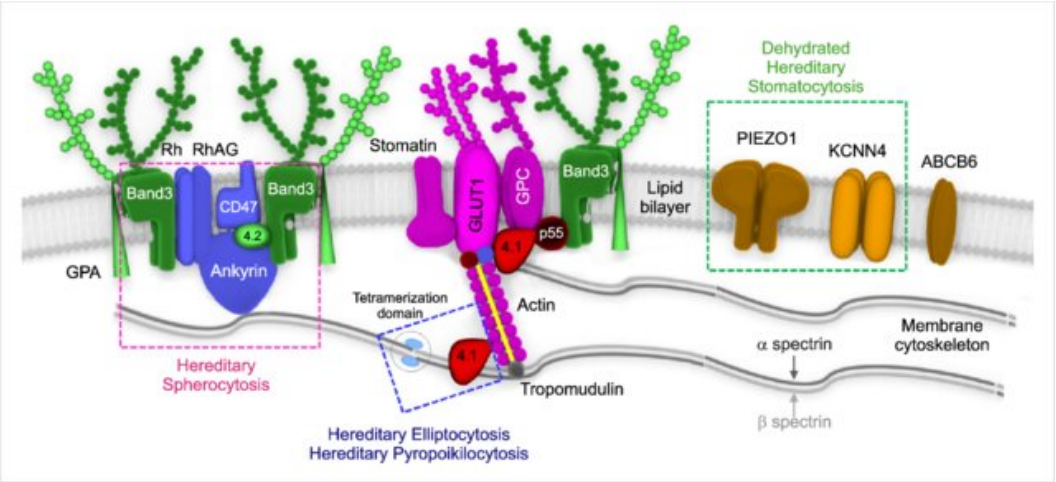


FIGURE 1.2: A schematic representation of RBC membrane structure with major functional components. The RBC membrane consists of three basic components: a lipid bilayer, transmembrane proteins, and a cytoskeletal network [2].

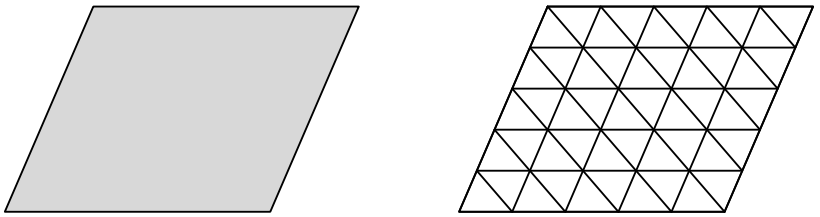
that exists between the PM and penetrates the cytosol interior [33]. It consists of a mesh-like network of mainly synthetic spectrin dimers. The spectrin dimers consist of α and β spectrin which are intertwined in parallel creating hetrodimers (see Figure 1.2). One end of the dimer is anchored to the PM via cytoskeleton proteins, while the other end is inter-connected to other dimers. This creates a hyper-elastic network which provides structural integrity to the RBC membrane. Of structural significance is that the relaxed shape of the cytoskeleton is considered to be a quasi-sphere and experiments indicate that the cytoskeleton tends to a near-spherical shape upon removal of the PM [34].

In summary the low viscosity ratio of the cytosol to the plasma, the lack of physical inter-constituent connections in the PM and the hyper-elastic network of the cytoskeleton are key aspects of the structure of a RBC. These properties enable the RBC to be a highly deformable and any RBC structural model should account for these properties.

1.5 Structural Modelling of a Red Blood Cell

There are different approaches to modelling the structural behaviour of a RBC with the main approaches being continuum approaches and discrete particle approaches (see Figure 1.3). What they have in common is the treatment of the membrane as a surface. The composite membrane of a RBC has a thickness of approximately

9-10 nm; this is orders of magnitude lower than the length scale of microns for a RBC. This allows the membrane to be treated as a two-dimensional (2D) surface embedded in three-dimensional (3D) space.



(a) (b)

FIGURE 1.3: Comparison of two RBC structural modelling approaches. (A) shows the continuum approach where the solid is modelled as a continuous surface. (B) shows the discrete particle approach where the solid is modelled by a network of particles interconnected with springs.

1.5.1 Continuum Red Blood Cell Models

The continuum approach predicts the structural mechanics of a solid by applying constitutive laws which define energy-strain relationships. Many laws have been suggested to model the in-plane mechanics of biological membrane. Skalak et al. [35] proposed a relationship where the material can deform in shear but conserves membrane surface area by specifying a very high area-dilation modulus. In comparison the Mooney-Rivlin approach considers the membrane to be a hyper-elastic incompressible solid [36]. This approach assumes an incompressible surface thickness which preserves the surface area of the membrane. A special-case of the Mooney-Rivlin approach is the neo-hookean law. The neo-hookean membrane is analogous to a Mooney-Rivlin membrane that has a negligible surface thickness [37]. Skalak's membrane is a strain-hardening material where as the Mooney-Rivlin and the neo-hookean membrane are strain-softening materials. However when deformations are considered to be small, all three membranes behave as Hooke's law which can be used to implement a Hookean membrane [38]. There is also a range of laws which have been proposed to model the out-of-plane bending resistance of the RBC membrane. The classical approaches of Canham [39] and Helfrich [40] proposed a resistance to local bending in a solid. Further proposals were made to account for non-local bending resistance due to the area-difference

between the outer and inner layer of the lipid bilayer. These include the "strict bilayer couple" model [41, 42] and the area-difference-energy model [43–45].

Continuum RBC models have been successfully implemented in two-phase blood flow problems to model a variety of RBC dynamic behaviours. These include tank-treading, tumbling, and swinging dynamics in simple shear flows [46–51]. Continuum approaches have also shown the slipper and parachute shapes encountered in capillary flow [52, 53]. They can also predict multi-cell behaviours such as the formation of rouleaux aggregates and cell free layers [49, 54]. More recent efforts have modelled the flow of RBCs in a Coulter counter [55]. However despite the accuracy of the continuum RBC approach, there are some limitations. As a large area dilation modulus is required to conserve membrane surface area, this can lead to significant numerical instabilities [56, 57]. Efforts have been made to remove this limitation such as implicit time integration; however this can only be used if cell lysis does not occur [58, 59]. Other efforts to resolve this limitation include the ad-hoc surface area dilation method of Pozrikidis [53] and using a Lagrange multiplier to enforce membrane incompressibility[60]. This effort to mitigate the numerical instability comes at a cost of computational expense [61]. A further limitations include an inability to capture the thermal fluctuations on the RBC membrane [62, 63].

1.5.2 Spring-Particle Red Blood Cell Models

Spring-network models can be used to model the solid mechanics of a surface. Boal [34] first suggested their application to model the solid mechanics of a RBC membrane. This approach involves discretising the surface into particles which are interconnected with springs. The mechanical properties of these springs then directly influence the behaviour of the model. The most common type of spring used in biological membranes are worm like chains (WLC) to model the in-plane shear resistance of the membrane [64]. Similar to the continuum approach, there are also differing approaches to modelling the out-of-plane bending resistance of the RBC. These include the spontaneous curvature model [64] and the area-difference-energy approach [65], with the latter accounting for non-local bending resistance due to the area-difference between the outer and inner layer of the lipid bilayer.

The earliest attempts at modelling RBC membrane mechanics looked at spring-networks at the spectrin level [64, 66, 67]. This resulted in accurate predictions

that compared favourably with experimental data and predictions of continuum structural solvers. However this approach is extremely computationally expensive with upwards of 20,000 vertices required to model individual spectrin dimers and tetramers. More recently coarse grained spring-particle (CG-SP) models have been developed where particles in the spring-network represent spectrin oligomers and the springs representing actin junction complexes [68–72]. This vastly reduces the computational effort while maintaining accuracy of the RBC mechanics with some studies using under 1,000 vertices [63]. Similar to the continuum models, the CG-SP approach has successfully modelled the RBC mechanics in two-phase flow. This includes tank-treading, tumbling, and swinging dynamics in simple shear flows [63, 69]. Slipper and parachute shapes encountered in capillary flow were also accurately modelled [63, 69, 73, 74]. Multi-cell behaviours such as the formation of rouleaux aggregates [75] and cell free layers [61] have also being modelled successfully. An advantage of the CG-SP methods over continuum methods is the ability to implement thermal fluctuations and diseased RBC dynamics [76–78].

1.5.3 Summary

The continuum and spring-particle methods are both proven approaches to modelling the behaviour of RBCs and two-phase blood flow. However the CG-SP approach has demonstrated enhanced modelling capabilities with its ability to thermal fluctuations and diseased RBC dynamics. The area-difference-energy approach to bending resistance is shown to account for the non-local bending resistance of the lipid bilayer. Most crucial is that the work of Chen and Boyle [65] is fully available and is adopted in this work to reduce development time.

1.6 Two-Phase Modelling of Blood Flow

The structural mechanics of RBCs have been described in the previous section. To model blood flow, both the RBC and the hydrodynamics of plasma need to be modelled. Plasma behaves like a Newtonian fluid and its behaviour can be predicted by solving the Navier-Stokes (N-S) equations. To fully model the behaviour of blood flow, the fluid-structure-interaction (FSI) between the plasma and the RBC needs to be modelled too. There are many approaches in the literature but the prominent approaches can be put into three groups: Boundary

Element Methods (BEM), unified plasma/RBC solvers and separate plasma/RBC solvers. These are discussed in the following sections.

1.6.1 Boundary Element Methods

The BEM approach utilises a boundary integral formation to create an analytical solution for the velocity field in the fluid [79]. The RBC is treated like a Lagrangian object in the fluid and from the boundary integral formation, the velocities at each of the nodes on the membrane are known. However the BEM can only be utilised when the effects of inertia are considered negligible. This reduces the N-S equations to the Stokes equations for this problem and this has many advantages, *i.e* no mesh is required to model the plasma and the computational expense required to calculate the velocity field is much reduced from a full N-S solver. The BEM has successfully been used to model two-phase blood flow including modelling the deformation of a RBC in shear flow [80–82] and pressure driven flow of RBCs in microcapillaries [53]. However further applications are limited by its Stokesian flow requirement.

1.6.2 Unified Plasma/RBC Solvers

Unified plasma/RBC solvers for predicting blood flow behaviour involve modelling the fluid and the RBC as a collection of particles that impart forces on each other as they move. At the heart of this family of methods is the idea that each particle represents a base object on the microscopic or mesoscopic level. Depending on the method, the particle can represent an atom, molecule or a portion of the fluid/RBC. RBCs structural mechanics are then modelled by adding interconnecting springs between the designated RBC particles. The lowest level approach of this family of methods is a molecular dynamics simulation where each particle represents a molecule or an atom [83]. However modelling at this scale is computationally impractical when real world hydrodynamic problems are considered. This lead to the development of mesoscopic and macroscopic level particle methods including dissipative particle dynamics (DPD), multi-particle collision dynamics (MPCD), smoothed particle hydrodynamics (SPH) and moving particle semi-implicit (MPSI) approach. A summary of these methods is provided

below. The reader is referred to Ye [84] for an in depth review of particle methods used to model blood flow.

The DPD method, first proposed by Hoogerbrugge and Koelmen [85], is essentially a coarse grained version of molecular dynamics. Here particles represent clusters of molecules and particles interact via conservative, dissipative and random forces. This approach has successfully been used to model a variety of blood flow problems including tank-treading, tumbling, and swinging dynamics in simple shear flows [63]. Slipper and parachute shapes encountered in capillary flow are also accurately modelled [63, 73, 74]. Multi-cell behaviours such as the formation of rouleaux aggregates [75] and cell free layers [61] have also been successfully modelled. Membrane viscosity, thermal fluctuations and diseased RBC dynamics have also been modelled [76–78].

The MPCD method again involves modelling the fluid and solid as coarse mesoscopic level particles. In this method mass, momentum and energy are conserved locally and flow is developed through alternating collision and streaming steps between the particles. An attractive feature of this method for modelling suspensions is that immersed particles can be coupled directly with the fluid by letting them participate in the streaming and collision steps [86]. It has been succesfully applied to model RBCs in shear and capillary flow and also the sedimentation of RBCs [69, 87, 88].

The SPH method discretises the N-S equations into a series of particles. A value of a parameter, e.g. velocity, is then calculated at a position by interpolating values of this parameter from neighbouring particles using a kernel function. Each particle moves in time according to Newton's second law. Hosseini and Feng [89] provide a detailed explanation of the forces acting on the particles. The SPH has been used to model RBCs in shear and capillary flow, and also to model malaria infected RBCs [89–91].

The MPSI method is very similar to the SPH method but uses implicit methods to integrate the position of each particle in time. It has been successfully used to model multi cellular flow in microvessels [92], malaria infected RBCs[93], margination of RBCs [94] and interaction between platelets and RBCs [95].

1.6.3 Separate Fluid/RBC Solvers

The third general approach to modelling two-phase blood flow is to use a general purpose N-S solver such as the Finite Volume Method (FVM), Finite Difference Method (FDM), Finite Element Methods (FEM) or Lattice Boltzmann Method (LBM) to predict the plasma flow and a structural solver to model the deformations of the RBC. The challenge then is to be able to model the FSI between the plasma and the RBC. All of these methods require that the velocity on the surface of the immersed object equals the velocity of the fluid at the same location and that the force on the surface is transferred to the fluid.

A simple approach is to use an arbitrary lagrangian-eulerian FEM [96] where the mesh continually adapts to changes in the geometry of the immersed object. However this leads to distinct challenges when complex geometries are involved and when immersed objects experience large deformations [97]. This approach also involves very costly remeshing and interpolation as the mesh is adapted on a continuous basis. A similar approach was also applied to the LBM [98] and used to model multi-cellular blood flow in micro vessels and platelet deposition [99, 100]. This approach involves finding links between lattice fluid nodes and the surface and applying standard bounce back conditions on these links to ensure the velocity at the surface equals the plasma velocity.

There are also a range of methods that don't require continuous adaptation of the mesh. These include the immersed boundary method (IBM), immersed finite element method (IFEM) and the fictitious domain method (FDM). The IBM was originally introduced by Peskin [101] for solving FSI problems with deformable objects and has been succesfully applied to blood flow. It includes modelling multi-cellular flow [71], arteriolar bifurcation [102], platelet margination [103], RBCs in Coulter counters [55], RBCs in shear flow [51] and malaria infected RBCs [104]. The IFEM was introduced by Zhang [105] and follows the same approach as the IBM while enabling the use of an unstructured FEM for modelling plasma. It has been used to model the aggregation of RBCs [106]. The FDM is again very similar but distributes a Lagrangian multiplier back to the fluid instead of a force [107]. It has also been used to model multi-cellular blood flow [108].

1.6.4 Summary

In summary there are a significant number of approaches to modelling two-phase blood flow which have been successfully applied to a range of applied blood flow problems. From the review above it can be concluded that the most mature approaches are the BEM, DPD and IBM. As the research motivation for this project is to investigate methods that are suitable for blood flow applications outside the microvasculature, the BEM is discounted due to its Stokesian flow requirement. It is also expected that one day this research could be utilised for biomedical device applications. These devices usually involve complex geometries such as heart valve leaflets. These devices also tend to have higher Reynolds number flows than in the circulatory system [109]. As a result unstructured grids are desirable to both capture the complex geometry of the devices and also provide refinement in the boundary layer of near-wall regions of the device. While efforts of introducing refinement [110] and arbitrary complex geometries [111] to DPD are documented in the literature, there is no successful application of both approaches together or evidence that these techniques have been succesfully applied to two-phase blood flow problems. In contrast the IBM has been succesfully used with a FVM approach on unstructured grids to a two-phase blood flow problem [51, 55, 112]. For this reason the IBM was adopted in this work.

1.7 Fluid Modelling Approaches

As the IBM has been chosen to couple the fluid and structure in the plasma, this allows the use of any N-S solver. A discussion of different approaches is given in the following sections:

1.7.1 Conventional Navier-Stokes Solvers

Conventional N-S solvers employ the FDM, FVM and FEM. They involve discretising a fluid into nodes, volumes and elements respectively and then predicting changes in macroscopic variables by approximating the N-S equations. The FDM approximates the derivative form of the N-S equations by calculating the differences between neighbouring nodes. The FVM approximates the integral form of

the N-S equations by calculating the flux of macroscopic properties between neighbouring cells. The FEM involves discretising a fluid into elements or cells. Then a parametric representation of unknown variables is calculated based on interpolation or shape functions defined across each element. Finally a weak formulation of the N-S equations is then solved using the parametric representation of the unknowns as inputs. Of the above methods only the FVM is fully conservative by default. It is also the method of choice in well established CFD codes such as Ansys Fluent and OpenFoam. For a detailed explanation on the FDM, FVM and FEM, the reader is referred to the following range of textbooks [113–116].

Further considerations are required when modelling incompressible flow using the FVM. The N-S equations solvers for incompressible flow can be split into two main categories: density based methods and pressure based methods. Both approaches require additional functionality to couple the density and velocity compared to compressible flow. Density based methods can be applied to incompressible flow by incorporating artificial compressibility techniques [117, 118]. Steady-state problems are solved by iterating to a steady-state solution with constant boundary conditions. Density based methods can be accelerated by using techniques such as Jacobian pre-conditioning, multigrid and residual smoothing [119]. Transient problems using artificial compressibility require the use of dual timestepping [120] where a solution is iterated in artificial time steps to steady state for each physical time step. Pressure based methods such as SIMPLE [121], SIMPLEC [122], PRIME [123] and PISO [124] involve an iterative procedure for solving the Poisson equation to calculate the pressure at each time step [115]. While both approaches allow an efficient approach to solving steady problems, the iterative procedure for each time step make both approaches computationally expensive for transient flow calculations [125].

1.7.2 Lattice Boltzmann Methods

In recent years the LBM has become a well established alternative for solving CFD problems [126–128]. It involves calculating the change in the density distribution of discrete particles at the mesoscopic level. The change in density distribution is due to the collision of the discrete particles and the subsequent streaming of the particles. Using a Chapman-Enskog expansion, it can be shown that the lattice Boltzmann equation (LBE) applied at a mesoscopic level is equivalent to the N-S

equations at a macroscopic level [129]. The LBM has many advantages compared with traditional N-S equations solvers when solving transient problems as it does not involve the solution of the expensive Poisson equation for pressure or iterative dual timestepping and uses a highly algebraic and parrellisable approach to calculating densities and velocities in a flow problem. Limitations of the standard LBM include the necessary use of a uniform Cartesian grid and the restriction of its use to flow problems with low Mach numbers. The required use of a uniform Cartesian grid is a major restriction when it comes to the modelling of real life engineering problems such as the haemodynamics of biomedical devices. Issues encountered include approximating curved boundaries using a staircase approximation and domain wide refinement of the grid to resolve boundary layer fluid dynamics. Using a staircase approximation introduces a geometrical discretisation error while domain wide refinement of the grid leads to excessive computational effort. This has resulted in much research into fully unstructured LBMs which would enable the use of body fitted grids with complex geometries and have local refinement near boundaries.

1.7.3 Unstructured Lattice Boltzmann Methods

The first attempt at an unstructured LBM was by Peng and Xi [130, 131]. This approach involved the integration of the differential form of the LBE in control volumes around the grid points. However it was found that the method suffered from significant instability issues. Over the years improvements to the stability of finite volume - lattice Boltzmann methods (FVLBM) have been made. Stiebler [132] introduced a least squares, linear reconstruction based upwind discretization scheme for the FVLBM. A total variation diminishing approach to the FVLBM was introduced by Patil [133] whereas Ubertini et al. [134] used a memory term to increase stability thresholds of the method. More recently Zarghami et al. [135] used upwind second-order pressure biasing factors as flux correctors to improve stability.

An alternative innovative approach was proposed by Wang et al. [136] and Shu et al. [137] when they developed the lattice Boltzmann flux solver (LBFS). This involves discretising the domain into cells and calculating the macroscopic fluxes at the cell interface using a local reconstruction of the LBE. Unlike in the FVLBMs

discussed above, the LBFS involves solving conservation equations for the macroscopic variables whereas the FVLBM involves solving conservation equations of the particle distributions. While the original method was implemented on non-uniform orthogonal meshes, lately it has been applied to fully unstructured tetrahedral meshes by Pellerin et al. [138] and Wu et al. [139]. It has also been extended to model FSIs on 3D geometries through the use of the immersed-boundary method by Wang et al. [140].

The LBFS shares many traits with the family of artificial compressibility methods (ACM) which was initially introduced by Chorin [117]. In the ACM an artificial relationship is introduced between pressure and density variables. This changes the nature of the governing equations from mixed elliptic/parabolic to hyperbolic/parabolic. This enables the use of efficient time-marching schemes in calculating the steady state solution of incompressible flows. The LBFS differs from the ACM in that the relationship between pressure and density is set by an arbitrary parameter in the ACM [117]. More recently efforts by Turkel [141] and Malan [142] have provided a means of calculating the optimum value of the arbitrary parameter. In the LBFS the relationship between pressure and density is defined by the lattice discretisation and there is no need to optimise the arbitrary parameter required in the ACM. The ACM and LBFS both solve the N-S equations, with the ACM using either a finite difference [143] or finite volume [142] spatial discretisation. The LBFS uses a finite volume approach. The ACM solves the N-S equations but uses a perturbed continuity equation and, as described by He et al [144], this has no physical meaning in incompressible flow. In contrast the LBFS is a "weakly" compressible method and its continuity equation has a physical meaning. There is also a difference in the stencils used in the calculation of the fluxes; the original ACM proposed a central differencing or leapfrog scheme [117] and more recently 3rd order upwind scheme and 4th order central compact schemes have been used [145]. In comparison the LBFS uses a local reconstruction of the LBE to calculate the fluxes. Further differences arise in that the ACM requires the use of artificial dissipation whereas the LBFS does not [143][142]. Also for unsteady flows the ACM is shown to require dual timestepping or similar implicit methods to effect real time accuracy [120] whereas the LBFS can effect real time accuracy using explicit and implicit methods. Finally while the original ACM can be thought of as a preconditioning approach to the continuity equation, this has also been extended to introduce preconditioning to the momentum equations [146]. This has the impact of improving convergence and robustness of the ACM approach.

1.7.4 Summary

The LBFS was adopted in this work due to its advantages over the FVLBM with regards to stability and the implementation of boundary conditions. The LBFS does not require the use of pressure biasing factors or other such schemes to achieve stability at higher Reynolds numbers. The LBFS also allows the direct implementation of physical boundary conditions whereas the FVLBM relies on the standard LBM family of boundary conditions which are more difficult to implement. It also retains advantages of the LBM of not requiring the solution of the Poisson equation or artificial time steps in transient flow, and having a localised computation which makes it highly parallelisable.

1.8 Runtime Acceleration

Blood flow is a very computationally expensive problem to model, with the number of RBCs per mm^3 of order $O(10^6)$ [147]. In practical engineering problems it is required to find a steady state solution as initial conditions for transient flow problems. The LBFS/LBM does not perform well in steady state calculations and as a result, methods of accelerating runtimes is required. Similarly to solve large scale blood flow problems it will be required to build a parrallisable and scalable code that makes use of modern day hardware. Both of these topics are discussed in the next two sections.

1.8.1 Preconditioning

A key characteristic of the standard LBM is the grid independent "compressibility" error which is directly proportional to the Mach number [148]. As a result LBM simulations limit this error by keeping the Mach number small; typically this involves keeping the Mach number less than 0.4 [128]. Reducing the Mach number has the knock on effect of increasing the disparity between the acoustic and convective wave speeds [149]. As the standard LBM typically employs explicit timestepping, the Courant-Friedrichs-Levy (CFL) condition should be satisfied to ensure stability. At low Mach numbers this requires a time step inversely proportional to the largest eigenvalue in the system which is approximately the speed of sound. However the convective wave propagates information through the domain

at the much lower fluid speed. As a result a large amount of time steps are required to reach steady-state convergence.

In recent years many researchers have made attempts to accelerate convergence of the standard LBM to steady-state convergence. A time independent formulation of the LBM has been proposed by Bernaschi et al. [150]. The steady-state solution is solved through iterative methods and this approach has also been implemented by Verberg [151] and Noble [152]. Tolke et al. [153] expanded on the time independent approach by using a multigrid approach to solve an implicit second-order finite difference scheme. An alternative approach for accelerating convergence of the LBM to steady-state is using implicit schemes to discretise the time-dependent equation. This allows larger time steps to be used and has been implemented by Lee [154], Seta [155] and Tolke et al. [156]. Further to this Mavriplis [157] proposed a non-linear form of multigrid solver with a non-linear LBE timestepping scheme. These methods all accelerate convergence of the LBM to steady-state but at the cost of increased complexity compared to the standard LBM.

Preconditioning is another time-dependent steady-state acceleration technique which was successfully applied to the LBM by Guo [158] originally. This approach follows the same principle as the preconditioning method developed by Turkel [149] to solve the incompressible and low speed compressible N-S equations. As mentioned above, at low Mach numbers there is a disparity between the acoustic and convective wave speeds. The ratio between these wave speeds is known as the condition number. The lower the condition number, the faster that a solution will converge to the steady-state solution. Preconditioning involves altering the eigenvalues of the N-S equations to reduce the condition number. Guo implemented preconditioning in the LBM by applying a single preconditioning factor (γ-preconditioner) to the equilibrium distribution function. For steady flows this results in an equivalent form of the N-S equations with a reduced condition number, which reduces the number of iterations required to reach steady-state.

There have been many additions to Guo's original work. Premnath et al. [159] extended the preconditioning approach to allow for forcing terms in force-driven fluid flow problems. Izquierdo et al. [160] extended it to the generalised form of the LBM including the multiple-relaxation time LBM. In this work a second preconditioning factor (β-preconditioner) was also used to improve the efficacy of preconditioning. They also investigated optimal values of the preconditioning values for the LBM and gave apriori guidelines for such values [161]. More recently

the approach has been extended to a noncascaded central moments LBM [162], a cascaded LBM [163] and a Galilean invariant cascaded LBM [164]. Meng et al. have also used an improved preconditioned multiple-relaxation time LBM to model flow through porous media [165].

In this work the γ-preconditioning approach of Guo's work was adopted as it has similar performance to the β-preconditioning approach of Izquierdo's work whilst being more efficient and also simpler to implement.

1.8.2 General Purpose Graphics Processing Unit Programming

Moore's law is widely interpreted as "the number of transistors on an integrated circuit would double every few years" [166]. Until approximately the year 2000, this has resulted in the 1.5x speed up in single threaded performance year on year (see Figure 1.4). However in recent years, the year-on-year gain in single threaded performance has decreased to approximately 1.1x speedup and the maximum frequency of chips has flat-lined. While the transistor count on microprocessors are still following exponential growth, the benefit in this is seen in a proportional increase in the number of cores in a microprocessor. The consequence of this is that any high performance code looking to take advantage of future advances in hardware must incorporate parallel computing [167].

Modern hardware that utilises many cores includes graphics processing unit (GPU) and many integrated core (MIC) chips. Both take advantage of having multiple cores with multiple threads running on each core but at lower clock speeds than high end central processing units (CPUs). This allows them to perform far more computations than single CPUs. An example of a GPU manufacturer is NVIDIA (Santa Clara, California, United States) who manufacture the Tesla GPUs for high performance computing (HPC). The flagship HPC is the Tesla Volta V100 which has a double precision processing power of 7.5 tera floating point operations per second (TFLOPS). GPUs tend to have host code which operates on the CPU and device code which is then run on the GPU. This requires an architecture to enable this host-device realationship. NVIDIA cards can be programmed for example using OpenCL, an open-source framework for programming GPUs, or CUDA, NVIDIA's proprietary API for programming NVIDIA GPUs. An example of MIC

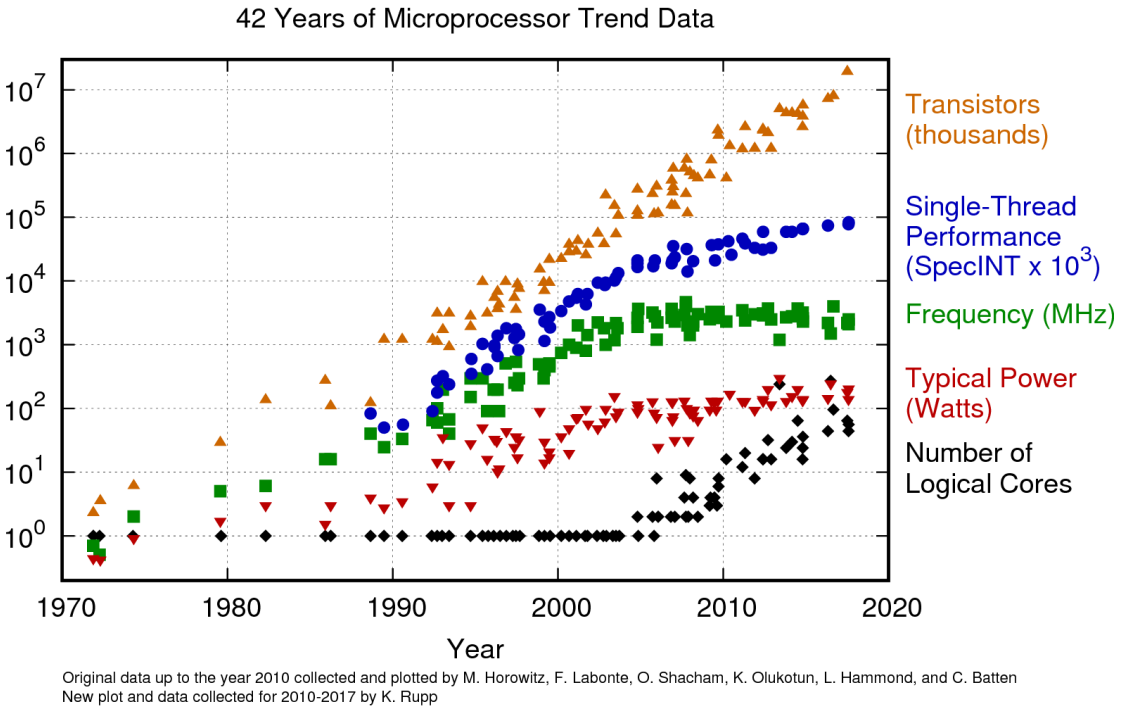


FIGURE 1.4: 42 years of microprocessor trends data. [3]

chip is Intels Xeon Phi series of chips. The Xeon Phi 7290 can achieve double precision processing power of 3.4 TFLOPS. The benefits of MIC is that there is no requirement for additional framework and architectures to produce code for the MIC. However an extensive knowledge of optimisations is required to fully utilise MIC [168]. While there is evidence in the literature of both GPUs and MICS being more advantageous than the other in certain scenarios, Teodoro et al. [169] provide evidence that GPUs perform better on operations with random data access and atomic operations. These are both characteristics of algorithms that involve unstructured grids like that encountered in LBFS N-S solvers and RBC spring-networks. For this reason and the increased maturity of NVIDIA GPUs, CUDA GPU programming was adopted in this work to parallelise the workflow.

1.9 Aim of Research

The aim of this research is to create the foundations of a blood flow solver that can handle the complex geometries required by biomedical applications. This involves using the LBFS to model plasma, an enhanced area-difference-energy spring-particle (ADE-SP) RBC model and the IBM to couple the fluid-structure

interactions. It will also aim to accelerate the runtime of the problem by utilising preconditioning in steady state fluid problems and utilise CUDA graphics processing units programming to parallelise the workflow.

1.10 Areas of Novelty

This research makes the following novel contributions to the literature:

- Development of a LBFS that utilises unstructured hexahedrals in 3D. Recent works have only extended the LBFS to non-uniform grids or 2D unstructured tetrahedral meshes.
- Development of a GPU accelerated LBFS.
- Implementation and investigation of preconditioning in a 3D unstructured LBFS.
- Implementation of IBM with a LBFS on non-uniform grids. Recent works have used the IBM with the LBFS exclusively with uniform meshes.
- Implementation, verification and validation of ADE-SP RBC model with a LBFS. SP RBC models have never been combined with the LBFS before.
- Development of a GPU accelerated ADE-SP LBFS blood flow model.

1.11 Outline of Dissertation

This chapter has outlined the existing state of the art in blood flow modelling in the literature. It has also presented the rationale behind key decisions made during this work and the objectives of this project. The rest of the dissertation presents the following work:

- Chapter 2 will introduce the governing equations related to the LBFS. This project did not have a LBFS code base to begin with and it was required to develop code in house. This process involves running numerical simulations of benchmark flow problems of increasing complexity. It can be verified that

the LBFS is correctly coded by comparing the results of the numerical simulations performed against benchmark results in the literature or analytical solutions. Each flow problem challenged a different aspect of the code and increased the confidence in the code developed to date.

- Chapter 3 contains the numerical experiments performed in 2D to verify the LBFS.
- Chapter 4 contains numerical experiments in 3D on unstructured hexahedral grids; it contains results to demonstrate the efficacy of preconditioning when applied to the LBFS.
- Chapter 5 introduces different approaches for implementing the IBM and also contains numerical experiments to verify the efficacy of the IBM approach adopted in this work.
- Chapter 6 contains the governing equations of the ADE-SP RBC structural model and the configurations adopted in this work.
- Chapter 7 contains numerical experiments which verify the RBC code is correctly implemented. The numerical predictions are validated against experimental measurements in the literature.
- Chapter 8 describes the key design decisions in implementing the ADE-SP LBFS blood flow model on the GPUs and also provides benchmarks for the runtime acceleration achieved.
- Finally, conclusions and recommendations for future work are presented in Chapter 9.

1.12 Publications

To date the work has led to publications in two peer-reviewed journals:

"A Preconditioned Lattice Boltzmann Flux Solver for Steady Flows on Unstructured Hexahedral Grids Computers and Fluids", Walsh B., Boyle F., Computers and Fluids

"Red Blood Cell Dynamics on Non-Uniform Grids Using the Lattice Boltzmann Flux Solver", Walsh B., Boyle F., TBD

The work of this project was also presented at the following conferences:

- Sir Bernard Crossland Symposium 2017.
- INSPIRE Conference - TU Dublin 2019.

Chapter 2

Lattice Boltzmann Flux Solver

2.1 Introduction

This chapter will provide a brief introduction to the governing equations of fluid flow using the LBFS. This includes the 3D N-S equations, the method of spatial discretisation and the evaluation of fluxes using the LBE. It will also introduce the non-dimensionalisation procedure, time-integration approach and a stability analysis.

2.2 Governing Equations

2.2.1 Navier-Stokes Equations

If body forces are neglected, the 3D unsteady N-S equations can be written in conservative form for a finite control volume in a Cartesian coordinate system as:

$$\frac{\partial}{\partial t}\int_V (\mathbf{Q}dV) + \int_S \mathbf{F}\cdot\mathbf{n}\; dS = 0 \tag{2.1}$$

where t is time, V is the cell volume, S is the cell surface, $\mathbf{n}$ is the outward normal to element of area dS, $\mathbf{Q}$ is the vector of conserved variables $[\rho, \rho u, \rho v, \rho w]$, ρ is the density, and u, v and w are the components of the velocity vector $\mathbf{V}$ in the x, y and z directions respectively. Assuming a Newtonian, isotropic, isothermal fluid

allows the flux tensor $\mathbf{F}$ to be defined as follows:

$$\mathbf{F} = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p - \mu\left(2\frac{\partial u}{\partial x}\right) & \rho uv - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \rho uw - \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \rho vu - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \rho v^2 + p - \mu\left(2\frac{\partial v}{\partial y}\right) & \rho vw - \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \rho wu - \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) & \rho wv - \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) & \rho w^2 + p - \mu\left(2\frac{\partial w}{\partial z}\right) \end{bmatrix} \tag{2.2}$$

where p is the static pressure and μ is the dynamic viscosity. The above equations can be used to simulate incompressible flow, when the Mach number is low and density variation is small.

Using a cell-centred finite volume approach for spatial discretisation, Equation (2.1) is applied to every cell in the computational domain yielding a set of semi-discrete equations:

$$\frac{d\mathbf{Q}_i}{dt} + \frac{1}{V_i}\sum_{n=1}^{N_{faces}} \mathbf{F}_{i,n} \cdot \mathbf{n}_{i,n} S_{i,n} = 0 \tag{2.3}$$

where for cell i, $\mathbf{Q}_i$ is the vector of conserved variables, V_i is the cell volume, N_{faces} is the number of cell faces, $\mathbf{n}_{i,n}$ is the outward normal of face n, $S_{i,n}$ is the area of face n, and $\mathbf{F}_{i,n}$ is the flux tensor at the centroid of face n.

2.2.2 Lattice Boltzmann Flux Solver

In the LBFS the flux tensor at each cell interface is evaluated from a local reconstruction of the lattice Boltzmann solution (LBS) at the cell interface. The cell interface is defined as the centroid of the shared face between the two adjoining cells. In this work the single relaxation time Bhatnagar-Gross-Krook (BGK) collision model is employed [170]. This leads to the following formulation for the flux tensor $\mathbf{F}$:

$$\begin{aligned} \mathbf{F} = &\sum_{\alpha=0}^{N} f_\alpha^{eq} \begin{bmatrix} e_{\alpha,x} & e_{\alpha,y} & e_{\alpha,z} \\ e_{\alpha,x}e_{\alpha,x} & e_{\alpha,y}e_{\alpha,x} & e_{\alpha,z}e_{\alpha,x} \\ e_{\alpha,x}e_{\alpha,y} & e_{\alpha,y}e_{\alpha,y} & e_{\alpha,z}e_{\alpha,y} \\ e_{\alpha,x}e_{\alpha,z} & e_{\alpha,y}e_{\alpha,z} & e_{\alpha,z}e_{\alpha,z} \end{bmatrix} \\ &+ \sum_{\alpha=0}^{N} \left[1 - \frac{1}{2\tau_s}\right] f_\alpha^{neq} \begin{bmatrix} 0 & 0 & 0 \\ e_{\alpha,x}e_{\alpha,x} & e_{\alpha,y}e_{\alpha,x} & e_{\alpha,z}e_{\alpha,x} \\ e_{\alpha,x}e_{\alpha,y} & e_{\alpha,y}e_{\alpha,y} & e_{\alpha,z}e_{\alpha,y} \\ e_{\alpha,x}e_{\alpha,z} & e_{\alpha,y}e_{\alpha,z} & e_{\alpha,z}e_{\alpha,z} \end{bmatrix} \end{aligned} \tag{2.4}$$

where $e_{\alpha,x}$, $e_{\alpha,y}$ and $e_{\alpha,z}$ are the x, y and z components respectively of the particle velocity vector $\mathbf{e}_\alpha$ in the α direction, τ_s is the standard relaxation factor, f_α is the density distribution function in the α direction, f_α^{eq} is the equilibrium density distribution function in the α direction, f_α^{neq} is the non-equilibrium density distribution function in the α direction and is equal to $f_\alpha - f_\alpha^{eq}$, and N is the number of velocities in the lattice model.

To implement the LBS at the cell interface, one must first choose a lattice velocity set to define $\mathbf{e}_\alpha$. Velocity sets are usually denoted in DdQq form where d denotes the number of spatial dimensions covered by the velocity set and q is the number of velocities in the set. In this work the D3Q15 velocity set is chosen due to its computational efficiency and is shown in Table 2.1. **i**, **j** and **k** are the unit vectors

	α	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\mathbf{e}_\alpha$	$c\cdot$ **i**	0	1	-1	0	0	0	0	1	-1	1	-1	1	-1	-1	1
	$c\cdot$ **j**	0	0	0	1	-1	0	0	1	-1	1	-1	-1	1	1	-1
	$c\cdot$ **k**	0	0	0	0	0	1	-1	1	-1	-1	1	1	-1	1	-1

TABLE 2.1: Explicit representation of the D3Q15 velocity set.

in the x, y and z directions respectively, $c = \delta x/\delta t$, δx is the lattice spacing along each Cartesian axis and δt is the streaming time step. Typically δx is set equal to δt giving c equal to 1 and this approach is adopted in this work. In physical terms c is considered to be the lattice velocity. When the D3Q15 lattice velocity set is applied to the cell interface, the lattice is formed by 15 nodes. The Cartesian coordinates of these lattice nodes can be represented in terms of lattice velocities. A 3D illustration of the D3Q15 lattice velocity set implemented at a cell interface is shown in Figure 2.1. In this figure the Cartesian coordinates of the two cell centres are referred to as $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ respectively. The cell interface is referred to as $\mathbf{r}$.

The LBE with the BGK collision model [170] can be applied to model a Newtonian fluid at the cell interface. This enables the finding of the equilibrium and non-equilibrium density distribution functions required to calculate the flux tensor. The LBE is given by:

$$\begin{aligned} f_\alpha(\mathbf{r},t) =& f_\alpha(\mathbf{r}-\mathbf{e}_\alpha\delta t, t-\delta t) + \\ & \frac{f_\alpha^{eq}(\mathbf{r}-\mathbf{e}_\alpha\delta t, t-\delta t) - f_\alpha(\mathbf{r}-\mathbf{e}_\alpha\delta t, t-\delta t)}{\tau_s} \quad for\ \alpha = 0, 1...N \end{aligned} \tag{2.5}$$

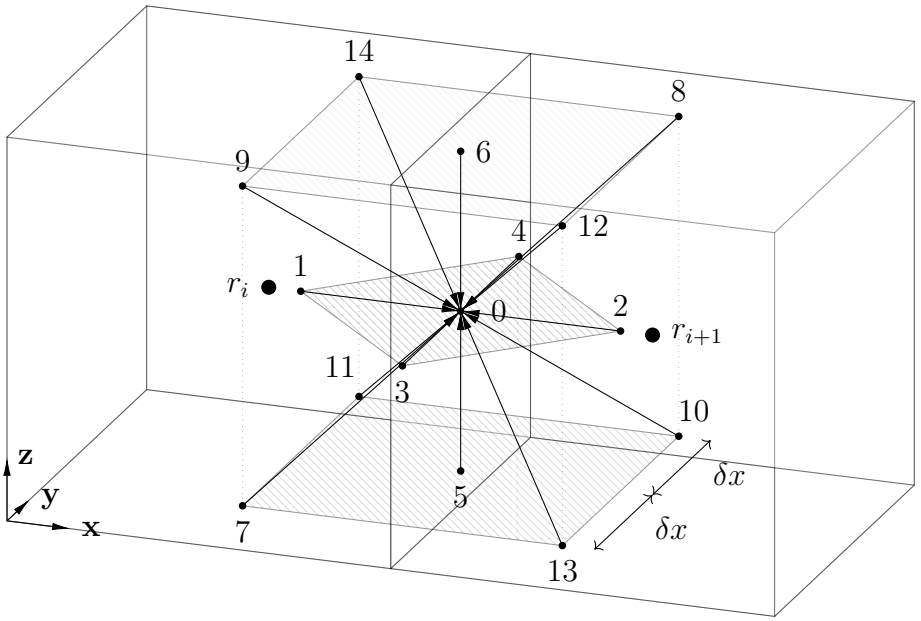


FIGURE 2.1: Local reconstruction of the LBS at a cell interface implementing a D3Q15 lattice velocity set.

where the standard relaxation factor τ_s is related to the viscosity and controls the influence of the viscous fluxes on the momentum equation. In physical terms it is the rate at which f_α tends to the equilibrium density distribution function f_α^{eq} and is given by:

$$\tau_s = \frac{\nu}{c_s^2 \delta t} + 0.5 \tag{2.6}$$

where ν is the kinematic viscosity and, as per the literature, c_s is the speed of sound in the lattice grid equalling $c/\sqrt{3}$ for a D3Q15 lattice model. The equilibrium density distribution function f_α^{eq} is given by a Hermite series expansion of the Maxwell-Boltzmann distribution:

$$f_\alpha^{eq}(\mathbf{r}, t) = \rho w_\alpha \left[1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{V}}{c_s^2} + \frac{(\mathbf{e}_\alpha \cdot \mathbf{V})^2 - (c_s|\mathbf{V}|)^2}{2c_s^4} \right] + O(\mathbf{V}^3) \tag{2.7}$$

where the weights w_α are given as:

$$w_\alpha = \begin{bmatrix} 2/9 \\ 1/9 \\ 1/72 \end{bmatrix} \quad \begin{matrix} \alpha = 0 \\ \alpha = 1 - 6 \\ \alpha = 7 - 14 \end{matrix} \tag{2.8}$$

As shown by Shu et al. [140], by using the Chapman-Enskog expansion and the linear Taylor series expansion of f_α^{eq}, the non-equilibrium density distribution

function f_α^{neq} can be calculated as follows:

$$f_\alpha^{neq}(\mathbf{r},t) = -\tau_s \left[f_\alpha^{eq}(\mathbf{r},t) - f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t) \right] \tag{2.9}$$

In the localised reconstruction of the LBE at the cell interface, the flux tensor is now dependent on the post-streaming and pre-streaming equilibrium density distribution functions $f_\alpha^{eq}(\mathbf{r},t)$ and $f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ respectively. These in turn are dependent on the density and velocity values at each lattice node. As f_α is a density distribution function, the density can be calculated by summing f_α up over all lattice velocities and similarly the momentum can be calculated by summing up the first moment of f_α, *i.e*:

$$\begin{aligned} \rho &= \sum_{\alpha=0}^{N} f_\alpha \\ \rho \mathbf{V} &= \sum_{\alpha=0}^{N} f_\alpha \mathbf{e}_\alpha \end{aligned} \tag{2.10}$$

From the equation of state, calculated through the Chapman-Enskog expansion, the pressure can be found using:

$$p = \rho c_s^2 \tag{2.11}$$

When a gas has been left alone for a sufficiently long period of time, it is assumed that f_α will reach an equilibrium which is defined by f_α^{eq}. We can make this assumption as collisions tend to even out the angular distribution of particle velocities in a gas around a mean velocity. As this convergence to equilibrium must conserve mass and momentum at all locations, equilibrium values can also be used to calculate the macroscopic density and momentum at any location as follows:

$$\begin{aligned} \rho &= \sum_{\alpha=0}^{N} f_\alpha^{eq} \\ \rho \mathbf{V} &= \sum_{\alpha=0}^{N} f_\alpha^{eq} \mathbf{e}_\alpha \end{aligned} \tag{2.12}$$

The pre-streaming equilibrium density distribution function $f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ is simply calculated by interpolating the macroscopic values from neighbouring cells

at time $t - \delta t$:

$$\rho(\mathbf{r} - \mathbf{e}_\alpha \delta t) = \left\{ \begin{array}{c} \rho(\mathbf{r}_i) + (\mathbf{r} - \mathbf{e}_\alpha \delta t - \mathbf{r}_i) \cdot \nabla \rho(\mathbf{r}_i) \\ \rho(\mathbf{r}_{i+1}) + (\mathbf{r} - \mathbf{e}_\alpha \delta t - \mathbf{r}_{i+1}) \cdot \nabla \rho(\mathbf{r}_{i+1}) \end{array} \right\} \begin{array}{c} \text{when } \mathbf{r} - \mathbf{e}_\alpha \delta t \text{ in cell } i \\ \text{when } \mathbf{r} - \mathbf{e}_\alpha \delta t \text{ in cell } i+1 \end{array} \tag{2.13}$$

$$\mathbf{V}(\mathbf{r} - \mathbf{e}_\alpha \delta t) = \left\{ \begin{array}{c} \mathbf{V}(\mathbf{r}_i) + (\mathbf{r} - \mathbf{e}_\alpha \delta t - \mathbf{r}_i) \cdot \nabla \mathbf{V}(\mathbf{r}_i) \\ \mathbf{V}(\mathbf{r}_{i+1}) + (\mathbf{r} - \mathbf{e}_\alpha \delta t - \mathbf{r}_{i+1}) \cdot \nabla \mathbf{V}(\mathbf{r}_{i+1}) \end{array} \right\} \begin{array}{c} \text{when } \mathbf{r} - \mathbf{e}_\alpha \delta t \text{ in cell } i \\ \text{when } \mathbf{r} - \mathbf{e}_\alpha \delta t \text{ in cell } i+1 \end{array} \tag{2.14}$$

Inputting these values into Equation (2.7) gives $f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$. The next step is to find a value for $f_\alpha^{eq}(\mathbf{r}, t)$. This is simply done by finding $\rho(\mathbf{r}, t)$ and $\mathbf{V}(\mathbf{r}, t)$. As mass and momentum are conserved $\rho(\mathbf{r}, t)$ and $\mathbf{V}(\mathbf{r}, t)$ can be calculated at the cell interface by summing the pre-streaming equilibrium distribution functions, *i.e.*:

$$\begin{aligned} \rho(\mathbf{r}, t) &= \sum_{\alpha=0}^{N} f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t) \\ \rho(\mathbf{r}, t)\, \mathbf{V}(\mathbf{r}, t) &= \sum_{\alpha=0}^{N} f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t) \mathbf{e}_\alpha \end{aligned} \tag{2.15}$$

Inputting these values of $\rho(\mathbf{r}, t)$ and $\mathbf{V}(\mathbf{r}, t)$ into Equation (2.7) will give $f_\alpha^{eq}(\mathbf{r}, t)$. Once $f_\alpha^{eq}(\mathbf{r}, t)$ and $f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ have been found, $f_\alpha^{neq}(\mathbf{r}, t)$ can be calculated and the fluxes at the cell interface calculated using Equation (2.4).

2.2.3 Boundary Conditions

In this work, the 'ghost cells' approach is adopted [113, 171]. This involves generating a fictitious neighbouring cell on all boundary faces in the computational domain. At the start of every time step the macroscopic variables are updated in the ghost cell dependent on the boundary condition. The macroscopic variables are chosen such that the required physical values at the boundary hold. For Dirichlet boundary conditions this means an explicit value of density or velocity at the boundary. For Neumann boundary conditions this means the specification of a gradient across the boundary. In contrast Shu et al. [137] explicitly calculate the boundary fluxes from the N-S equations. On unstructured grids, the

approach described by Moukallad [115] is required to remove error introduced by non-Cartesian boundary faces. Once the macroscopic value of the 'ghost cell' is specified, the flux across the boundary can be calculated the same as any other face in the computational domain.

2.2.4 Gradient Calculation

A key element of the LBFS is the accurate calculation of the gradient of the macroscopic variables at each cell centre as these gradients are then used in Equation (2.13) and Equation (2.14) to initialise the local LBS at the cell interface. An inaccurate calculation of the gradient will lead to a local LBS that does not reflect the flow field correctly. Two prominent methods used to calculate gradients on unstructured grids are the Green-Gauss and Least-Squares methods. Shima et al. [172] note how the Green-Gauss method is only fully accurate on uniform and symmetric grids but performs better on the thin and curved meshes with high aspect ratio cells that often exist within boundary layers for high Reynolds number flow calculations. Shima proposed a new hybrid Green-Gauss/Weighted-Least-Squares (GLSQ) approach for calculating the gradient. The GLSQ involves a geometry-dependent switch which applies the Green-Gauss method to those cells with a high aspect ratio and the Least-Squares method to cells with a low aspect ratio. This approach also has the benefits of ensuring monotonicity at cell interfaces, increasing the robustness of the solver, and allowing the accurate handling of hanging nodes in the gradient calculation. This is for a small increase in computational cost compared to the Least-Squares method. For these reasons the GLSQ approach is adopted in this work and is explained in detail in A.4.

2.3 Non-Dimensionalisation Procedure

The LBM is predominantly solved in terms of "lattice units". This is where the physical variables in the simulation are represented by dimensionless numbers in terms of the lattice units. The process involves converting the physical variables into non-dimensional form and then discretising these parameters in terms of the lattice units in the lattice velocity model chosen. A detailed analysis and proof of the non-dimensionalisation procedure is given in Appendix A.3. This section

will summarise the non-dimensionalisation procedure used in the LBFS and also provide a brief note on variable selection for incompressible flow modelling.

2.3.1 Non-Dimensionalisation Procedure for LBFS

Let * indicate a non-dimensional variable. The following is the non-dimensionalisation procedure for the LBFS approach:

1. First it is decided that the non-dimensional lattice spacing δx^* and streaming step δt^* are equal to one. The justification for this is that it reduces the number of calculations required in the stream and collision process.

2. Next the following non-dimensional procedure is implemented:

$$x^* = \frac{x}{\delta x} \qquad y^* = \frac{y}{\delta x} \qquad z^* = \frac{z}{\delta x}$$
$$u^* = \frac{u}{\delta x/\delta t} \qquad v^* = \frac{v}{\delta x/\delta t} \qquad w^* = \frac{w}{\delta x/\delta t}$$
$$\rho^* = \frac{\rho}{\rho_{ref}} \qquad t^* = \frac{t}{\delta t}$$

   where δx is the lattice spacing in the physical problem, δt is the streaming time step in the physical problem, and ρ_{ref} is the reference density.

3. Applying this procedure to the N-S equations leads to the following form of the equation:

$$\frac{d}{dt^*}\int\int_{V^*}(\mathbf{Q}^* dV^*) + \int_{S^*}\mathbf{F}_I^* \cdot \mathbf{n}\; dS^* + \int_{S^*}\mathbf{F}_V^* \cdot \mathbf{n}\; dS^* = 0 \tag{2.16}$$

   See Appendix A.3 for details of the proof.

4. $\mathbf{F}_I^*$ and $\mathbf{F}_V^*$ are the inviscid and viscous flux contributions respectively populated by non-dimensional parameters:

$$\mathbf{F}_I^* = \sum_{\alpha=0}^{N} f_\alpha^{eq} * \begin{bmatrix} e_{\alpha,x}^* & e_{\alpha,y}^* & e_{\alpha,z}^* \\ e_{\alpha,x}^* e_{\alpha,x}^* & e_{\alpha,y}^* e_{\alpha,x}^* & e_{\alpha,z}^* e_{\alpha,x}^* \\ e_{\alpha,x}^* e_{\alpha,y}^* & e_{\alpha,y}^* e_{\alpha,y}^* & e_{\alpha,z}^* e_{\alpha,y}^* \\ e_{\alpha,x}^* e_{\alpha,z}^* & e_{\alpha,y}^* e_{\alpha,z}^* & e_{\alpha,z}^* e_{\alpha,z}^* \end{bmatrix} \tag{2.17}$$

$$\mathbf{F}_V^* = \sum_{\alpha=0}^{N} \left[1 - \frac{1}{2\tau_s}\right] f_\alpha^{neq} * \begin{bmatrix} 0 & 0 & 0 \\ e^*_{\alpha,x}e^*_{\alpha,x} & e^*_{\alpha,y}e^*_{\alpha,x} & e^*_{\alpha,z}e^*_{\alpha,x} \\ e^*_{\alpha,x}e^*_{\alpha,y} & e^*_{\alpha,y}e^*_{\alpha,y} & e^*_{\alpha,z}e^*_{\alpha,y} \\ e^*_{\alpha,x}e^*_{\alpha,z} & e^*_{\alpha,y}e^*_{\alpha,z} & e^*_{\alpha,z}e^*_{\alpha,z} \end{bmatrix} \tag{2.18}$$

5. The next step is to discretise the non-dimensional form of the N-S equations in terms of the lattice grid in the LBM. The lattice velocities can be calculated from Table 2.1, f_α^{eq*} can be calculated from Equation (2.7) and f_α^{neq*} can be calculated from Equation (2.9) using non-dimensional forms of time, position and density. All that remains to be calculated is the relaxation factor τ^*.

6. The law of similarity as explained by Landau and Lifshitz [173] states:

   > For incompressible flow, two flow systems are dynamically similar so long as the geometry and Reynolds number are the same.

   This means that:

   $$Re^* = Re \tag{2.19}$$

7. Assuming a uniform grid, as shown in Appendix A.3, the relaxation factor τ^* can be calculated as:

   $$\tau^* = 0.5 + \frac{3\sqrt{3}Ma^* N_{Cells}\Delta x^*}{Re^*} \tag{2.20}$$

   where Ma^* is the Mach number of the lattice model, N_{Cells} is the number of cell volumes along the reference length, Δx^* is the length of each cell and Re^* is the Reynolds number of the lattice model. Ma^* and Re^* are given by:

   $$Ma^* = \frac{\mathbf{V}^*_{max}}{c^*_s} \tag{2.21}$$

   $$Re^* = \frac{\rho^* \mathbf{V}^*_{max} D^*}{\mu^*} \tag{2.22}$$

   where $\mathbf{V}^*_{max}$ is the maximum velocity in the lattice model and D^* is the reference length of the simulated problem in terms of lattice units.

8. For completeness the non-dimensional viscosity can be calculated as:

   $$\nu^* = (\tau^* - 0.5)[c^*_s]^2 \delta t = \frac{\sqrt{3}Ma^* N_{Cells}\Delta x^*}{Re^*} \tag{2.23}$$

NOTE: For the rest of the thesis, the non-dimensional notation is dropped and all variables are considered to be in their non-dimensional form.

2.3.2 Guidance on Parameter Selection for Incompressible Flow on Uniform Grids

It is also required for incompressible flow to perform diffusive scaling when refining the mesh density. Dropping the non-dimensional notation and using non-dimensional variables. This means that δt scales $\propto \delta x^2$. Without this scaling, it has been shown that incompressible flow calculations will not converge [174]. This is due to compressibility errors generated by the lack of third order term in the equilibrium distribution function (see Equation (2.7)). This error is of the order $O(\mathbf{V}^3)$. This is in comparison to the spatial discretisation error and the time discretisation error which have errors of $O(\delta x^2)$ and $O(\delta t^2)$ respectively. Without diffusive scaling, the compressibility error remains constant and drowns out the solution, resulting in instability. The main points are summarised above but a more in depth discussion is provided in Appendix A.3.

2.4 Time Integration of the Navier-Stokes Equations

One advantage of the LBFS is that it allows a decoupling of the lattice streaming time step and the time integration of the N-S equations. This enables the use of many different integration methods for time-integration of the N-S equations in the LBFS.

There are many time integration techniques available. These can be mainly categorised into implicit and explicit methods. Implicit methods have the advantage of being unconditionally stable and allow large time steps to be applied. While the allowable time step size is theoretically infinite, it is limited by the order of accuracy required of the time integration method. However, this comes at a computational cost as the CPU cost per iteration is significantly higher than explicit methods. Generally speaking this is due to the need for matrices to be inverted at each time step for implicit methods. Explicit methods on the other hand are

conditionally stable and tend to have very small allowable time steps due to stability criteria. As no matrices are required to be inverted with explicit methods, the CPU cost per time step is lower than with implicit methods.

When choosing a method, it is important to look at the time scales of the physical problem that is being modelled. If this is of the order of the stability limit of an explicit method, then it is advisable to use explicit methods as the computational cost per time step is lower. As this research is investigating transient blood flow with individual deformations of RBCs, there is a requirement of integrating in time using small time steps to ensure a high order of accuracy. Therefore, it was decided that an explicit scheme should be used.

One family of methods is the Runge-Kutta explicit integration schemes. These allow an arbitrarily high order of accuracy in time and higher order methods are well adapted to the stability criteria of centrally discretised convection problems [113]. As shown by Jameson et al. [175], the fourth-order Runge-Kutta (RK4) scheme is shown to be a very efficient and robust time integration approach for a finite volume discretisation. For these reasons the RK4 scheme was chosen as the time integration method for the LBFS.

The rest of this section will describe the RK4 scheme, its stability region and the associated stability criteria.

2.4.1 Four Stage Runge-Kutta Scheme

Assuming the area of cells in a problem remain independent of time, partitioning the computational domain into contiguous cells and applying Equation (2.1) to each cell gives:

$$\frac{d\mathbf{Q}}{dt} + \frac{1}{V}\sum_{n=1}^{N_{faces}} \mathbf{F}_n \cdot \mathbf{n}_n S_n = 0 \tag{2.24}$$

Equation (2.24) can be rewritten as:

$$\frac{d\mathbf{Q}}{dt} + \mathbf{R(Q)} = 0 \tag{2.25}$$

where $\mathbf{R(Q)}$ is the residual function defined by:

$$\mathbf{R(Q)} = \frac{1}{V}(\mathcal{L}_I + \mathcal{L}_V)\mathbf{Q} \tag{2.26}$$

$\mathcal{L}_I$ is the inviscid flux spatial discretisation operator and $\mathcal{L}_V$ is the viscous flux spatial discretisation operator defined as follows:

$$\mathcal{L}_I = \int_S \mathbf{F}_I \cdot \mathbf{n} \, dS \tag{2.27}$$

$$\mathcal{L}_V = \int_S \mathbf{F}_V \cdot \mathbf{n} \, dS \tag{2.28}$$

A RK4 integration scheme can be typically described as:

$$\begin{aligned}
\mathbf{Q}^{(0)} &= \mathbf{Q}^{(t)} \\
\mathbf{Q}^{(1)} &= \mathbf{Q}^{(0)} - \frac{\Delta t}{2}\mathbf{R}^{(0)} \\
\mathbf{Q}^{(2)} &= \mathbf{Q}^{(0)} - \frac{\Delta t}{2}\mathbf{R}^{(1)} \\
\mathbf{Q}^{(3)} &= \mathbf{Q}^{(0)} - \Delta t\mathbf{R}^{(2)} \\
\mathbf{Q}^{(t+1)} &= \mathbf{Q}^{(0)} - \frac{\Delta t}{6}(\mathbf{R}^{(0)} + 2\mathbf{R}^{(1)} + 2\mathbf{R}^{(2)} + \mathbf{R}^{(3)})
\end{aligned} \tag{2.29}$$

where $\mathbf{R}^{(q)} = \mathbf{R}(\mathbf{Q}^{(q)})$ for $q = 0$ to 4, Δt is the time integration time step and t denotes the t^{th} time step in the simulation.

2.4.2 RK4 Stability Region

When looking to predict the stability of a numerical scheme, there will be stability criteria driven by the spatial discretisation and also stability criteria driven by the time integration method. For an overall numerical scheme to be stable, the time integration method must be compatible with the spatial discretisation. In this section the stability zone of the RK4 method will be discussed.

First rewrite Equation (2.29) in terms of a numerical amplification factor g:

$$\mathbf{Q}^{(t+1)} = g\mathbf{Q}^{(t)} \tag{2.30}$$

This allows any solution for a given time step t to be rewritten in terms of the initial solution $\mathbf{Q}^{(0)}$ at $t = 0$ as:

$$\mathbf{Q}^{(t+1)} = g^{(t+1)}\mathbf{Q}^{(0)} \tag{2.31}$$

g is a constant which is dependent on a single parameter $\Delta t\lambda$ where λ is an unknown variable for now. The stability of the solution requires $\mathbf{Q}^{(t)}$ to remain bounded. This will occur if g^t remains bounded for all t and Δt. This gives the requirement that $|g| < 1$ for $t \longrightarrow \infty$ and for all values of Δt. The residuals in Equation (2.29) can be rewritten in terms of the parameter $\Delta t\lambda$:

$$\begin{aligned}
\mathbf{R}^{(0)} &= \Delta t\lambda \mathbf{Q}^{(t)} \\
\mathbf{R}^{(1)} &= \Delta t\lambda \left(\mathbf{Q}^{(t)} + \frac{\Delta t\lambda \mathbf{Q}^{(t)}}{2} \right) \\
\mathbf{R}^{(2)} &= \Delta t\lambda \left(\mathbf{Q}^{(t)} + \frac{\Delta t\lambda \left(\mathbf{Q}^{(t)} + \frac{\Delta t\lambda \mathbf{Q}^{(t)}}{2} \right)}{2} \right) \\
\mathbf{R}^{(3)} &= \Delta t\lambda \left(\mathbf{Q}^{(t)} + \frac{\Delta t\lambda \left(\mathbf{Q}^{(t)} + \frac{\Delta t\lambda \left(\mathbf{Q}^{(t)} + \frac{\Delta t\lambda \mathbf{Q}^{(t)}}{2} \right)}{2} \right)}{2} \right)
\end{aligned} \tag{2.32}$$

After much algebra [176], Equation (2.32) can be shown to be:

$$\mathbf{Q}^{(t+1)} = \left(1 + \Delta t\lambda + \frac{\Delta t\lambda^2}{2} + \frac{\Delta t\lambda^3}{6} + \frac{\Delta t\lambda^4}{24} \right) \mathbf{Q}^{(t)} \tag{2.33}$$

This gives g as:

$$g = \left(1 + \Delta t\lambda + \frac{\Delta t\lambda^2}{2} + \frac{\Delta t\lambda^3}{6} + \frac{\Delta t\lambda^4}{24} \right) \tag{2.34}$$

For a RK4 scheme to converge $|g|$ must be less than 1. Letting λ be a complex number $a + bi$ where $i = \sqrt{-1}$ and substituting into Equation (2.34) gives g in complex form. After a substantial amount of algebra the real and imaginary parts of g can be calculated as:

$$Real(g) = 1 + b + \frac{b^2 - a^2}{2} + \frac{b^3 - 3a^2 b}{6} + \frac{a^4 + b^4 - 6a^2 b^2}{24} \tag{2.35}$$

$$Img(g) = ai + abi + \frac{3b^2 ai - a^3 i}{6} + \frac{4b^3 ai - a^3 i}{6} + \frac{4a^3 ai - 4a^3 bi}{24} \tag{2.36}$$

. The complementary values of $Real(g)$ and $Img(g)$ for which $|g| < 1$ were calculated numerically. See Appendix A.6 for the code that calculated the contour plot where $|g| < 1$. The resulting stability region is shown in Figure 2.2.

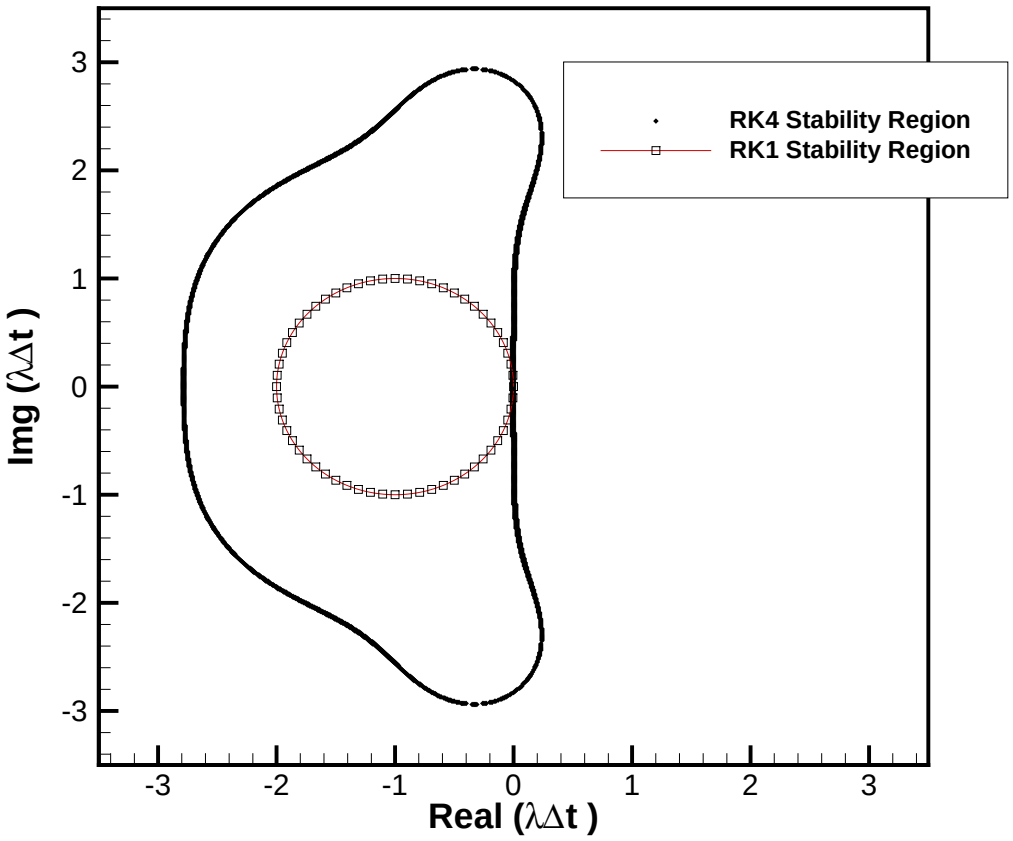


FIGURE 2.2: RK4 stability region.

2.4.3 RK4 Stability Criteria

With this stability region, the stability criteria for RK4 schemes can now be calculated. The stability criteria are that the real and imaginary parts of $\Delta t\lambda$ must lie within the stability region shown in Figure 2.2. This results in:

$$\begin{aligned} -2.78 < Real(\Delta t\lambda) < 0 \\ -2.83i < Img(\Delta t\lambda) < 2.83i \end{aligned} \tag{2.37}$$

But what is λ? As described by Sowa [177] $Real(\lambda)$ are the eigenvalues of the viscous flux Jacobian and $Img(\lambda)$ are the eigenvalues of the inviscid flux Jacobian. How to calculate these eigenvalues will be discussed in Section 2.5.

2.5 Stability Analysis of the Navier-Stokes Equations

It has been shown in Section 2.4 that the stability and time step of the LBFS is dependent on the stability region of the RK4 method and the Jacobians of the viscous and inviscid fluxes of the N-S equations. This section will show the values of these eigenvalues and hence show how to calculate the allowable time step for the LBFS.

2.5.1 Non-Conservative form of Navier-Stokes Equations

For the purposes of the stability analysis, the non-dimensional N-S equations in Equation (2.16) can be rewritten in non-conservative, symmetric, partial differential equation form [178] as:

$$\begin{aligned}\frac{\partial \mathbf{Q}}{\partial t} + A\frac{\partial \mathbf{Q}}{\partial x} + B\frac{\partial \mathbf{Q}}{\partial y} + J\frac{\partial \mathbf{Q}}{\partial z} = \\ + C\frac{\partial^2 \mathbf{Q}}{\partial^2 x} + D\frac{\partial^2 \mathbf{Q}}{\partial^2 y} + K\frac{\partial^2 \mathbf{Q}}{\partial^2 z} + E_{xy}\frac{\partial^2 \mathbf{Q}}{\partial x \partial y} + E_{yz}\frac{\partial^2 \mathbf{Q}}{\partial y \partial z} + E_{zx}\frac{\partial^2 \mathbf{Q}}{\partial z \partial x}\end{aligned} \tag{2.38}$$

where

$$\mathbf{Q} = \begin{bmatrix} \rho \\ u \\ v \\ w \end{bmatrix} \tag{2.39}$$

See Appendix A.5, Equations (A.38) to (A.44) for details of the symmetric coefficients A to E_{zx} in Equation (2.38). From Sowa [177], the Fourier transform of Equation 2.38 in 3D yields:

$$\frac{\partial \mathbf{Q}_\omega}{\partial t} = P_\omega \mathbf{Q}_\omega \tag{2.40}$$

where P_ω is the Fourier form of the numerical amplification operator and is given by:

$$P_\omega = -\left(iA\xi_x + iB\xi_y + iJ\xi_z + C\xi_x^2 + D\xi_y^2 + K\xi_z^2 + E_{xy}\xi_x\xi_y + E_{yz}\xi_y\xi_z + E_{zx}\xi_z\xi_x\right) \tag{2.41}$$

where $\xi_j = \frac{sin(\omega \Delta j)}{\Delta j}$, $j \in x, y, z$ for the discrete case and i is the imaginary i. $\frac{sin(\omega \Delta j)}{\Delta j}$ is bounded by $\frac{1}{\Delta j}$ in the discrete case. The inviscid part of Equation 2.41

for the discrete case is then given by:

$$P_{\omega,Inviscid} = -\left(iA\xi_x + iB\xi_y\right) + iJ\xi_z$$
$$= -i\begin{bmatrix} \frac{u}{\Delta x}+\frac{v}{\Delta y}+\frac{w}{\Delta z} & \frac{c_s}{\Delta x} & \frac{c_s}{\Delta y} & \frac{c_s}{\Delta z} \\ \frac{c_s}{\Delta x} & \frac{u}{\Delta x}+\frac{v}{\Delta y}+\frac{w}{\Delta z} & 0 & 0 \\ \frac{c_s}{\Delta y} & 0 & \frac{u}{\Delta x}+\frac{v}{\Delta y}+\frac{w}{\Delta z} & 0 \\ \frac{c_s}{\Delta z} & 0 & 0 & \frac{u}{\Delta x}+\frac{v}{\Delta y}+\frac{w}{\Delta z} \end{bmatrix} \tag{2.42}$$

The eigenvalues of $P_{\omega,Inviscid}$ (see Equation (2.42)) are:

$$\lambda_{Inviscid} = i\begin{bmatrix} \left(\frac{1}{\Delta x}\right)\left(|u|+|v|\frac{\Delta x}{\Delta y}+|w|\frac{\Delta x}{\Delta z}\right) \\ \left(\frac{1}{\Delta x}\right)\left(|u|+|v|\frac{\Delta x}{\Delta y}+|w|\frac{\Delta x}{\Delta z}\right) \\ \left(\frac{1}{\Delta x}\right)\left(|u|+|v|\frac{\Delta x}{\Delta y}+|w|\frac{\Delta x}{\Delta z}\right)+c_s^2\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}} \\ \left(\frac{1}{\Delta x}\right)\left(|u|+|v|\frac{\Delta x}{\Delta y}+|w|\frac{\Delta x}{\Delta z}\right)-c_s^2\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}} \end{bmatrix} \tag{2.43}$$

The viscous part of Equation (2.41) for the discrete case is then given by:

$$P_{\omega,Viscous} = -\left(C\xi_x^2 + D\xi_y^2 + K\xi_z^2 + E_{xy}\xi_x\xi_y + E_{yz}\xi_y\xi_z + E_{zx}\xi_z\xi_x\right) =$$
$$-i\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho}\left[\frac{2}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\right] & \frac{\mu}{\rho}\left[\frac{1}{\Delta x\Delta y}\right] & \frac{\mu}{\rho}\left[\frac{1}{\Delta x\Delta z}\right] \\ 0 & \frac{\mu}{\rho}\left[\frac{1}{\Delta x\Delta y}\right] & \frac{\mu}{\rho}\left[\frac{1}{\Delta x^2}+\frac{2}{\Delta y^2}+\frac{1}{\Delta z^2}\right] & \frac{\mu}{\rho}\left[\frac{1}{\Delta y\Delta z}\right] \\ 0 & \frac{\mu}{\rho}\left[\frac{1}{\Delta x\Delta z}\right] & \frac{\mu}{\rho}\left[\frac{1}{\Delta y\Delta z}\right] & \frac{\mu}{\rho}\left[\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{2}{\Delta z^2}\right] \end{bmatrix} \tag{2.44}$$

The eigenvalues of $P_{\omega,Viscous}$ (see Equation (2.44)) are:

$$\lambda_{Viscous} = -\left[\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right] \begin{bmatrix} \frac{2\mu}{\rho} \\ \frac{\mu}{\rho} \\ \frac{\mu}{\rho} \\ 0 \end{bmatrix} \tag{2.45}$$

2.5.2 Timestepping using Eigenvalues from Viscous and Inviscid Flux Vectors

As shown in Equation (2.37), the real and imaginary parts of the eigenvalues must lie within the RK4 stability region. The inviscid eigenvalues are strictly imaginary so they must comply with the constraint on the imaginary part of the RK4 stability region. The viscous eigenvalues are strictly real so they must comply with the constraint on the real part of the RK4 stability region. This results in the following constraints:

$$\begin{aligned} -2.83i < \lambda_{Inviscid}\Delta t < 2.83i \\ -2.78 < \lambda_{Viscous}\Delta t < 0 \end{aligned} \tag{2.46}$$

Substituting the largest eigenvalues from Equation (2.43) and Equation (2.45) into Equation (2.46) gives the following timestepping criteria for RK4 stability:

$$\lambda_{Inviscid,Max} = i\left(\frac{1}{\Delta x}\right)\left(|u| + |v|\frac{\Delta x}{\Delta y} + |w|\frac{\Delta x}{\Delta z}\right) + c_s^2\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}} \tag{2.47}$$

$$-2.83 < \left(\left(\frac{1}{\Delta x}\right)\left(|u| + |v|\frac{\Delta x}{\Delta y} + |w|\frac{\Delta x}{\Delta z}\right) + c_s^2\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}\right)\Delta t < 2.83 \tag{2.48}$$

$$\lambda_{Viscous,Max} = -\left(\frac{2\mu}{\rho}\left[\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right]\right) \tag{2.49}$$

$$-2.78 < -\left(\frac{2\mu}{\rho}\left[\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right]\right)\Delta t < 0 \tag{2.50}$$

In the case of a uniform grid where $\Delta x = \Delta y$, $\lambda_{Viscous,Max}$ becomes:

$$\lambda_{Viscous,Max} = -\left(\frac{3}{\Delta x^2}2\nu\right) \tag{2.51}$$

Substituting the LBM form of viscosity from Equation (2.23) into Equation (2.50) for a uniform grid gives:

$$\begin{aligned}\lambda_{Viscous,Max} &= -\left(\frac{2}{\Delta x^2}2\frac{\sqrt{3}Ma^*N_{Cells}\Delta x}{Re^*}\right) \\ &= -\left(\frac{4\sqrt{3}Ma^*N_{Cells}}{Re^*\Delta x}\right)\end{aligned} \tag{2.52}$$

Similarly, $\lambda_{Inviscid,Max}$ becomes:

$$\lambda_{Inviscid,Max} = i\left(\frac{1}{\Delta x}\right)(|u|+|v|+|w|) + c_s^2\sqrt{\frac{3}{\Delta x^2}} \tag{2.53}$$

The first thing to note is that all the terms in Equation (2.52) and Equation (2.47) are in non-dimensional form. Secondly, note that the above limits only apply where the flow is strictly convective or diffusive. In cases where both convective and diffusive behaviour occurs, the real and imaginary parts of the eigenvalues would have to be mapped in the stability region in Figure 2.2.

2.5.3 Eigenvalues – Swanson and Turkel Method

Swanson and Turkel [119] devised an alternative rule of thumb for calculating a time step which would ensure stability of the RK scheme. While it has no formal mathematical basis, in practice it has shown to be quite effective as it is a conservative estimate of the optimum time step. The formula for the general time step is as follows:

$$\frac{1}{\Delta t} = \frac{1}{\Delta t_{viscous}} + \frac{1}{\Delta t_{inviscid}} \tag{2.54}$$

2.5.4 Eigenvalues Approximation on Unstructured Grids

There are many ways to estimate the maximum allowable time step on unstructured grids with the following approach adopted in this work [179]:

$$\Delta t_i = \sigma\frac{V_i}{(\Lambda_c + C\Lambda_v)_i} \tag{2.55}$$

where Λ_c and Λ_v represent estimates of the spectral radii (i.e. largest absolute value of its eigenvalues) of the convective and viscous Jacobians respectively for

cell i. C is an arbitrary constant which is normally chosen to be $C = 4$ and V_i is the volume of cell i. In the case of a cell-centred finite volume scheme, the spectral radii are calculated as:

$$\begin{aligned} \Lambda_c &= \Lambda_{c,x} + \Lambda_{c,y} + \Lambda_{c,z} \\ \Lambda_v &= \Lambda_{v,x} + \Lambda_{v,y} + \Lambda_{v,z} \end{aligned} \tag{2.56}$$

where the Cartesian components of the spectral radii are given as:

$$\begin{aligned} \Lambda_{c,x} &= (|u| + c_s)\, S_{i,x} \\ \Lambda_{c,y} &= (|v| + c_s)\, S_{i,y} \\ \Lambda_{c,z} &= (|w| + c_s)\, S_{i,z} \\ \Lambda_{v,x} &= \frac{2\mu}{\rho} \frac{(S_{i,x})^2}{V_i} \\ \Lambda_{v,y} &= \frac{2\mu}{\rho} \frac{(S_{i,y})^2}{V_i} \\ \Lambda_{v,z} &= \frac{2\mu}{\rho} \frac{(S_{i,z})^2}{V_i} \end{aligned} \tag{2.57}$$

where $S_{i,x}$, $S_{i,y}$ and $S_{i,z}$ represent projections of the cell onto y-z, x-z and x-y planes respectively, *i.e.*:

$$\begin{aligned} S_{i,x} &= \frac{1}{2} \sum_{n=1}^{N_{faces}} |S_x|_{i,n} \\ S_{i,y} &= \frac{1}{2} \sum_{n=1}^{N_{faces}} |S_y|_{i,n} \\ S_{i,z} &= \frac{1}{2} \sum_{n=1}^{N_{faces}} |S_z|_{i,n} \end{aligned} \tag{2.58}$$

where S_x, S_y and S_z denote the x, y and z components of the face vector $\mathbf{S}_{i,n} = \mathbf{n}_{i,n} \cdot S_{i,n}$, $S_{i,n}$ is the surface area and $\mathbf{n}_{i,n}$ is the outward normal of face n of cell i. The time step for the whole domain is then given as:

$$\Delta t = min\,(\Delta t_i) \tag{2.59}$$

For unstructured meshes, where the ratio of the largest cell to the smallest cell can be large, this can result in extremely slow convergence of the solution to a steady-state. However the transient behaviour of the flow in steady problems is of no

interest. As a result local timestepping can be employed and each cell is progressed at its maximum allowable stable time step calculated by Equation (2.55). This results in significant convergence acceleration.

2.6 Summary

This chapter has introduced the governing equations of the dimensional and non-dimensional form of the LBFS. It has also introduced the time integration methods and stability criteria employed in this work. The computational procedure employed in the LBFS is summarised in Table 2.2.

	LBFS Algorithm
1.	Calculate the local time step for each cell using Equation (2.55) with $\mathbf{Q}^{(t-1)}$ as input.
2.	Calculate the gradients at the cell centres using Equation (A.29).
3.	At the cell interface initialise a D3Q15 local lattice Boltzmann solution with δx chosen so that all lattice grid nodes lie within the two cells adjoining the shared surface.
4.	Find $\rho(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ and $\mathbf{V}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ at the lattice grid nodes by interpolation using Equation (2.13) and Equation (2.14).
5.	Calculate the pre-streaming equilibrium density distribution function $f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ using Equation (2.7).
6.	Calculate $\rho(\mathbf{r}, t)$ and $\mathbf{V}(\mathbf{r}, t)$, the post-streaming macroscopic variables at the cell interface, from Equation (2.15).
7.	Use these values to calculate $f_\alpha^{eq}(\mathbf{r}, t)$ using Equation (2.7).
8.	$f_\alpha^{neq}(\mathbf{r}, t)$ can be calculated from $f_\alpha^{eq}(\mathbf{r}, t)$ and $f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)$ using Equation (2.9).
9.	The inviscid and viscous fluxes can then be calculated from $f_\alpha^{eq}(\mathbf{r}, t)$ and $f_\alpha^{neq}(\mathbf{r}, t)$ using Equations (2.17) and (2.18).
10.	The solution can then be advanced in time using Equation (2.29).

TABLE 2.2: A detailed summary of the LBFS solution procedure.

Chapter 3

Two-Dimensional Benchmark Flow Problems

3.1 Introduction

Without having a LBFS code base to begin with, it was required to develop code in house. This process involves running numerical simulations of benchmark flow problems of increasing complexity. It can be verified that the LBFS is correctly coded by comparing the results of the numerical simulations performed against benchmark results in the literature or analytical solutions. Each flow problem challenged a different aspect of the code and increased the confidence in the code developed. This chapter shows the 2D numerical experiments that were performed. These include:

- Couette flow.
- Poiseuille flow.
- Lid-driven cavity flow.
- Taylor-Green vortex flow.
- Womersley flow.
- Lid-driven cavity flow on non-uniform grids.

The initial verification was performed in 2D to prove that the LBFS is a viable candidate for solving blood flow problems in 3D. Working in 2D requires less development time, less computational resources and offers a broader range of analytical and benchmark solutions with which to verify the efficacy of the code. Once the LBFS was verified for a variety of benchmark problems in 2D, then the additional commitment in extending the LBFS to 3D could be justified. Note that all numerical experiments were performed on quasi 3D meshes *i.e.* a 3D mesh with a thickness of one cell in the Cartesian z direction.

3.2 Couette Flow

3.2.1 Introduction

Couette flow is a simple test case used to illustrate shear driven flow. It consists of two plates of infinite length and width, one of which is stationary and the other which is moving (see Figure 3.1).

In this case the N-S equations simplify to:

$$\frac{\partial^2 u}{\partial y^2} = 0 \tag{3.1}$$

Applying the following boundary conditions:

$$u\left(y=0\right)=0 \; and \; u\left(y=h\right)=U_{max} \tag{3.2}$$

where U_{max} is the velocity of the moving plate and h is the distance between the two plates, gives an exact solution to Equation (3.1) of:

$$u\left(y\right)=U_{max}\frac{y}{h} \tag{3.3}$$

3.2.2 Problem Set Up

The analytical solution of Couette Flow is for plates of infinite lengths. This was implemented in the computational domain by implementing periodic boundary condition for velocity at $x = 0$ and a Neumann boundary condition for velocity at

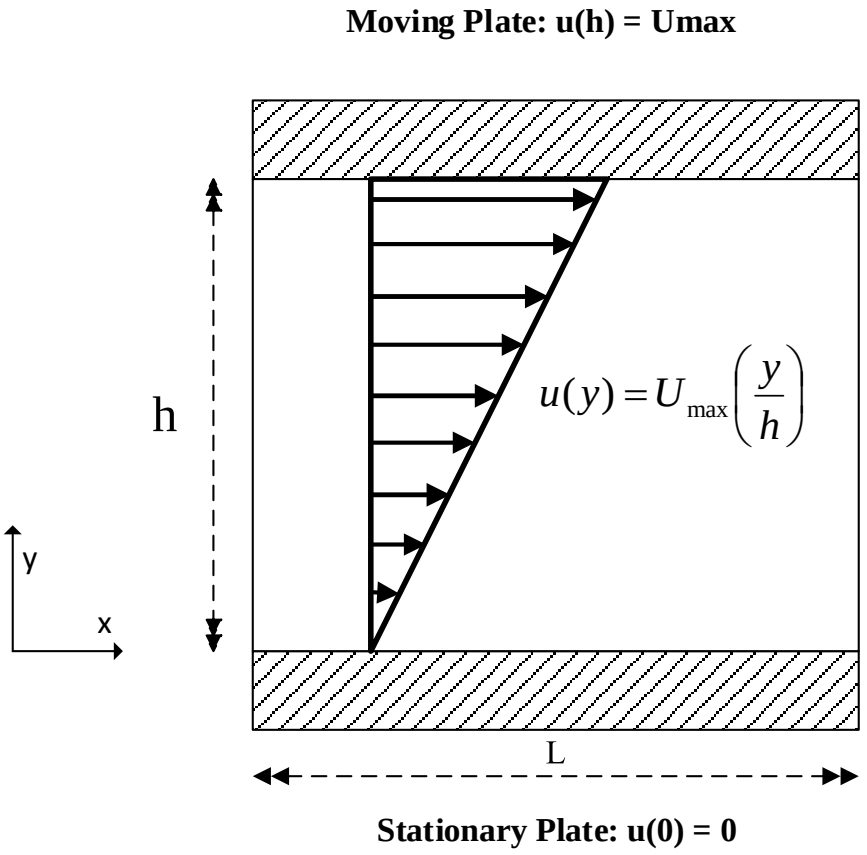


FIGURE 3.1: Couette flow - set up.

$x = L$. These can be summarised as:

$$u\,(x = 0, t) = u\,(x = L, t - 1) \;\; and \;\; \frac{\partial u(x = L)}{\partial x} = 0 \tag{3.4}$$

The flow problem was run using parameters that are similar to those required in a hemodynamic fluid flow problem. In this case viscosity was chosen as $1.275\,\mathrm{mPa\,s}$ as this is the average viscosity of Plasma at $37\,^{\circ}\mathrm{C}$ [21]. The largest diameter in normal coronary arteries has been shown to be in the range of $3.9\,\mathrm{mm} - 4.3\,\mathrm{mm}$ [180]. For fluid velocity, the average velocity of blood in the aorta is $40\,\mathrm{cm\,s^{-1}}$ [181] and the density of plasma is given at $37\,^{\circ}\mathrm{C}$ as $1.0205\,\mathrm{kg\,m^{-3}}$ [182]. Using these parameters results in a Reynolds number of approximately 1375. The above values are summarised in standard SI units in Table 3.3.

Individual test cases were then run for a variety of mesh densities. These are summarised in Table 3.2.

Variable Name	Value	Units
h	0.0043	m
μ	0.001275	Pa s
$u(h)$	0.4	$\mathrm{m\,s^{-1}}$
ρ	1020.5	$\mathrm{kg\,m^{-3}}$
Reynolds No.	1375	-

TABLE 3.1: Overall test case parameters for Couette flow.

Test Case No.	Mesh Density (No. of Cells)	$\frac{\Delta x}{\delta x}$	$\Delta t(s)$
1	15	2	0.000344
2	30	2	0.000172
3	60	2	8.59987E-05
4	120	2	4.29993E-05
5	240	2	4.29993E-05
6	480	2	1.07498E-05
7	960	2	5.37492E-06

TABLE 3.2: Individual test case parameters for Couette flow.

An illustration of the mesh used in Test Case 1 is given in Figure 3.2.

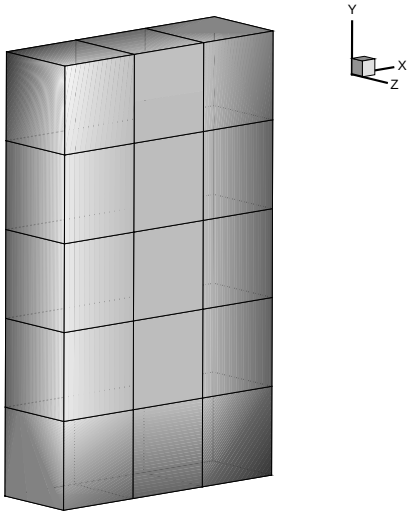


FIGURE 3.2: Couette flow - mesh for Test Case 1.

3.2.3 Convergence Criteria

For all test cases, a scaled residual was used to monitor convergence to steady state. The convergence criteria was defined as follows:

$$\frac{R\left(\mathbf{Q}_{Iteration,N}\right)}{R\left(\mathbf{Q}_{Iteration,1}\right)} < 10^{-9} \tag{3.5}$$

where:

$$R\left(\mathbf{Q}_{Iteration,N}\right) = \sqrt{\sum_{1}^{No\ of\ Cells} \left[\frac{d\mathbf{Q}}{dt}\right]} \tag{3.6}$$

3.2.4 Results

3.2.4.1 Numerical Solution Vrs Analytical Solution

The numerical solution plotted against the analytical solution for Test Case 3 is shown in Figure 3.3.

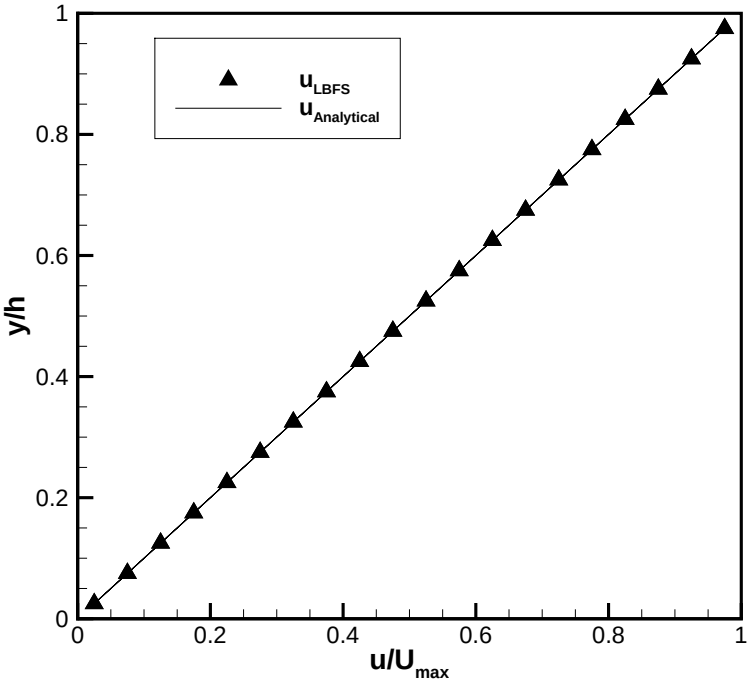


FIGURE 3.3: Couette flow - numerical solution vrs analytical solution.

3.2.5 Conclusions

The test case was successfully performed for a variety of test cases with the numerical solutions matching the analytical solution to a high degree of accuracy (see Figure 3.3). This demonstrates that the LBFS correctly calculates the viscous fluxes and that no-slip, moving, periodic and outlet boundary conditions have been implemented correctly.

3.3 Poiseuille Flow

3.3.1 Introduction

Poiseuille flow is a simple test case used to illustrate pressure driven flow. It consists of two stationary plates of finite length with a pressure differential between the inlet and outlet (see Figure 3.4).

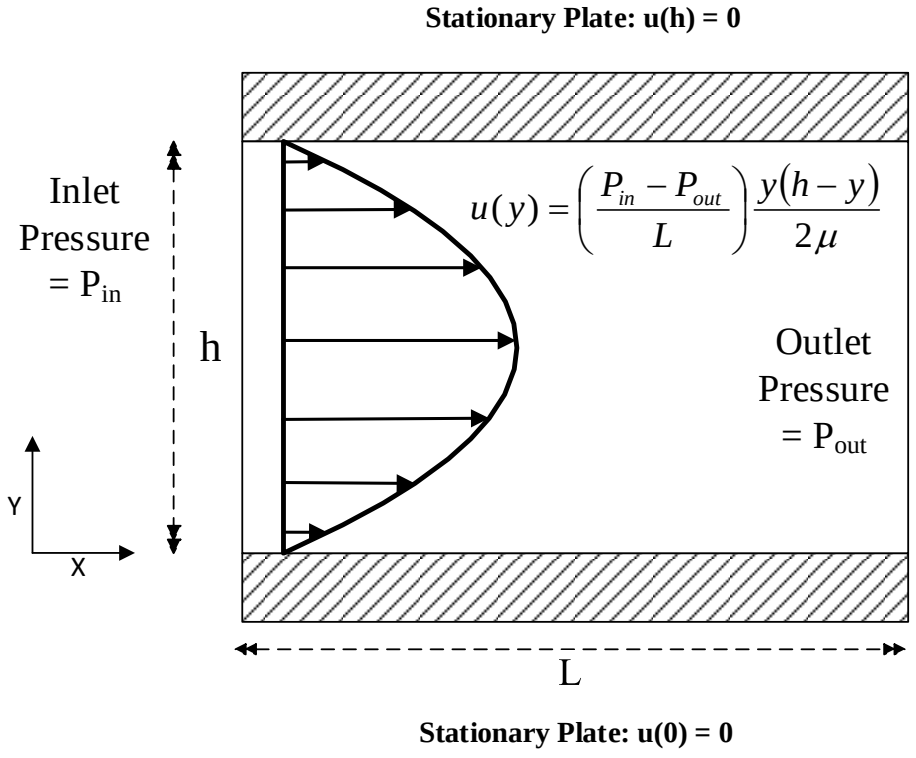


FIGURE 3.4: Poiseuille flow - set up.

An analytic solution can be derived which gives the following equation for the velocity profile:

$$u\left(y\right) = \frac{\Delta P}{L} \cdot \frac{y\left(h-y\right)}{2\mu} \tag{3.7}$$

3.3.2 Problem Set Up

The analytical solution of Poiseuille flow is for static plates of finite length and a pressure differential between the inlet and outlet. This was implemented in the computational domain by implementing a no slip boundary condition for velocity at $y = 0$ and $y = h$. In the LBFS pressure and density are related by Equation (2.11); the pressure differential was implemented by using Dirichlet boundary conditions for density at the inlet and outlet. These can be summarised as:

$$\rho\left(x=0,t\right) = \frac{P_{inlet}}{3} \; and \; \rho\left(x=L,t\right) = \frac{P_{outlet}}{3} \tag{3.8}$$

The flow problem was run using parameters that are similar to those required in a hemodynamic fluid flow problem. Values of viscosity of plasma, density of plasma, blood flow velocity and artery diameters were chosen in the same manner as in Section 3.2.2. From these values a realistic value for the pressure gradient can be calculated. This results in the following pressure gradient:

$$\frac{\Delta P}{\Delta x} = \frac{0.4 \cdot 8 \cdot 0.001275}{0.0043^2} = 220.658 \mathrm{Pa\,m^{-1}} \tag{3.9}$$

Using these parameters results in a Reynolds number of approximately 1375. The above values are summarised in standard SI units in Table 3.3.

Variable Name	Value	Units
h	0.0043	m
μ	0.001275	$\mathrm{Pa\,s^{-1}}$
$u(h/2)$	0.4	$\mathrm{m\,s^{-1}}$
ρ	1020.5	$\mathrm{kg\,m^{-3}}$
Reynolds No.	1375	-
$\frac{\Delta P}{\Delta x}$	221	$\mathrm{Pa\,m^{-1}}$

TABLE 3.3: Overall flow problem parameters for Poiseuille flow.

Individual test cases were then run for a variety of mesh densities. These are summarised in Table 3.4.

Test Case No.	Mesh Density (No. of Cells)	$\frac{\Delta x}{\delta x}$	$\Delta t(s)$
1	50	2	0.000344
2	100	2	0.000172
3	200	2	8.59987E-05
4	400	2	4.29993E-05
5	800	2	4.29993E-05
6	1600	2	1.07498E-05
7	3200	2	5.37492E-06

TABLE 3.4: Individual test case parameters for Poiseuille flow.

3.3.3 Convergence Criteria

The convergence criteria used were those described in Section 3.2.3.

3.3.4 Results

3.3.4.1 Numerical Solution Vrs Analytical Solution

The numerical solution plotted against the analytical solution for Test Case 7 is shown in Figure 3.5.

3.3.5 Conclusions

The test case was successfully performed for a variety of test cases with the numerical solutions comparing favourably to the analytical solution (see Figure 3.5). This demonstrates that the LBFS can correctly predict the outcome of pressure driven flow and that convective and viscous fluxes are accurately calculated for mono-directional flow.

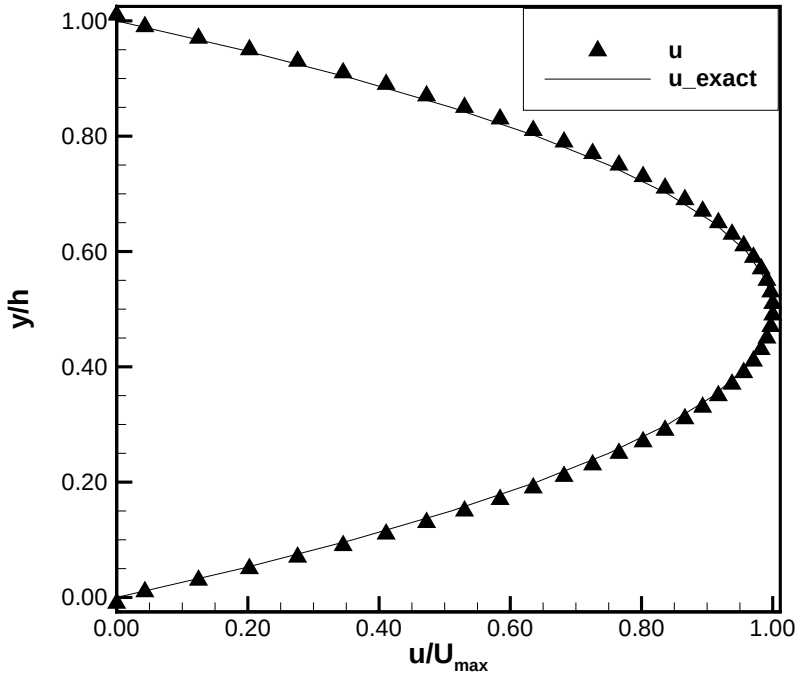


FIGURE 3.5: Poiseuille flow - numerical solution vrs analytical solution.

3.4 Lid-Driven Cavity

3.4.1 Introduction

Shear-driven flow in a cavity is a standard flow problem for validating incompressible viscous flow. It has historically been used as the benchmark study to compare the accuracy and the performance of CFD codes [183]. Lid-driven cavity flows also provide a model for understanding more complex flows with closed recirculation regions. Such regions can be found in blood flows over medical devices as well as in patients suffering from stenosis [184]. For these reasons, the 2D lid-driven cavity flow of an incompressible fluid was selected as a suitable benchmark flow problem for the LBFS.

3.4.2 Problem Set Up

The lid-driven cavity is a steady state problem at low Reynolds numbers with a moving lid developing the flow. The LBFS iterates the solution in time until a steady state is reached. This can then be compared to existing numerical results produced by Ghia et al. [4]. The initial set up of the lid-driven cavity problem is illustrated in Figure 3.6. The top boundary moves with a velocity u_{lid}. This was implemented using a Dirichlet boundary condition specifying u_{lid}. A no-slip boundary condition was implemented at the remaining three boundaries. This was implemented using a Dirichlet boundary condition specifying zero velocity at these boundaries. For pressure, a Neumann boundary condition specifying zero change in pressure across the boundary was implemented at all boundaries.

The fluid in the cavity was chosen as water due to its similar properties to plasma. The test case was run for a range of Reynolds numbers, *i.e.* Re = 100, 400, 1000, 3200, for the physical parameters provided in Table 3.15.

Variable Name	Value	Units
L_{ref}	0.1	m
ν	$1.0034 * 10^{-6}$	$\mathrm{m\,s^{-1}}$
ρ	998.2	$\mathrm{kg\,m^{-3}}$

TABLE 3.5: Overall test case parameters for lid-driven cavity.

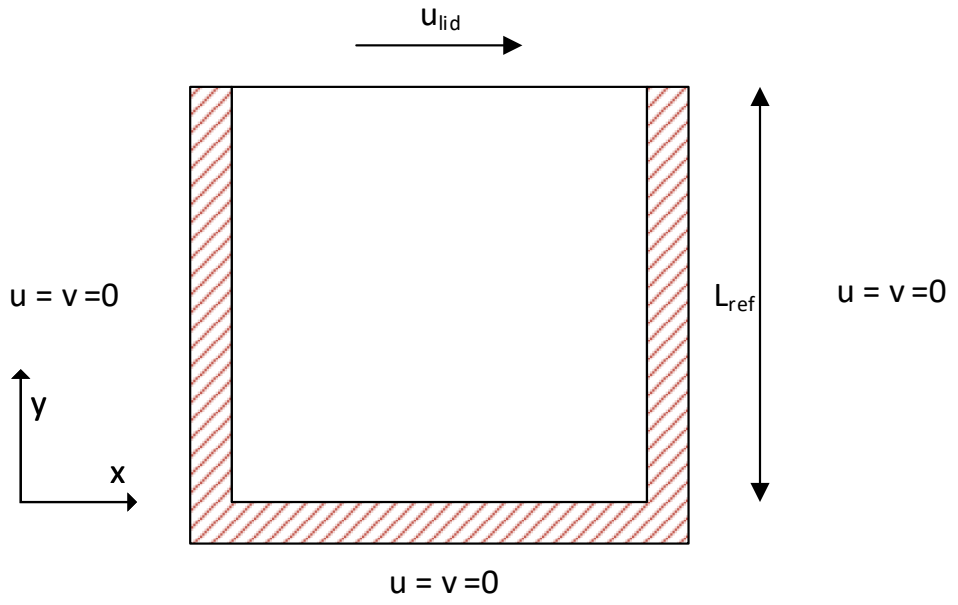


FIGURE 3.6: Lid-driven cavity - set up.

The value of velocity changes with Reynolds number and a test case was ran for each Reynolds number. The parameters for the test cases are summarised in Table 3.6. The test case for $Re = 100$ was run for two grid sizes. The first grid of $N_x = N_y = 50$ was done to give a comparison with the results achieved by Wang et al. [136]; the other grid size of $N_x = N_y = 129$ was to give a comparison with the results achieved by Ghia [4].

Test Case No.	Re	u_{lid} (m/s)	Mesh Density (No. of Cells)	$N_{x,y}$	$\frac{\Delta x}{\delta x}$	Δt (s)
1	100	0.001	2500	50	2	0.1
2	100	0.001	16641	129	2	0.03875
2	400	0.004	16641	129	2	0.00279
3	1000	0.010	16641	129	2	0.00187
4	3200	0.032	16641	129	2	0.00141

TABLE 3.6: Individual test case parameters varying with Reynolds number for lid-driven cavity.

3.4.3 Convergence Criteria

For all test cases, the root mean square (RMS) of the velocity change between successive iterations was used to monitor convergence to steady state. The convergence criteria was defined as follows:

$$R\left(\mathbf{Q}_{Iteration,N}\right) < 1 * 10^{-6} \tag{3.10}$$

where:

$$R\left(\mathbf{Q}_{Iteration,N}\right) = \sum_{1}^{No\ of\ Cells} \sqrt{\frac{\left(\left(\sqrt{u^2+v^2}\right)^{N+1} - \left(\sqrt{u^2+v^2}\right)^{N}\right)^2}{\left(\left(\sqrt{u^2+v^2}\right)^{N+1}\right)^2}} \tag{3.11}$$

This measure of residual was the same as used by Wang et al. [136].

3.4.4 Results

The results generated by the LBFS were compared to the numerical results produced by Ghia et al. [4]. The following aspects of the simulation are compared:

- Centre line velocity plots.
- Vorticity contours.
- Streamline plots.
- Primary vortex location.

The velocity profiles of the steady state solution were calculated across the centrelines of the domain in the x and y direction. The results for Test Cases 1-5 are shown in Figures 3.7 to 3.11.

Vorticity is a measure of the rotationality of the flow at a particular point in space. It is given mathematically by Equation (3.12):

$$\boldsymbol{\zeta} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)\mathbf{i} + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)\mathbf{j} + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\mathbf{k} \tag{3.12}$$

In 2D this reduces to:

$$\boldsymbol{\zeta} = \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \mathbf{k} \tag{3.13}$$

The vorticity contour plots for each of the test cases are plotted and a comparison with the results of Ghia et al. [4] is provided in Figures 3.12 to 3.16. The normalised values of the vorticity for each of the contours is provided in Table 3.7.

Contour Number	Value of ζ
0	0
± 1	± 0.5
± 2	± 1.0
± 3	± 2.0
± 4	± 3.0
5	4.0
6	5.0

TABLE 3.7: Normalised vorticity values for contour numbers in Figures 3.12 to 3.16.

A streamline is a curve that is everywhere tangent to the instantaneous velocity vector. It can alternatively be defined by the streamfunction ψ. Curves of constant ψ are streamlines of the flow. The streamfunction can be calculated numerically from the vorticity by solving the following Poisson equation:

$$\nabla^2 \psi = -\boldsymbol{\zeta} \tag{3.14}$$

The streamline plots for each of the test cases are plotted and a comparison with the results of Ghia et al. [4] is provided in Figures 3.17 to 3.21.

The location of the centre of the primary vortex is the part with the minimum value of the streamfunction. It is then compared to results provided by Ghia et al. [4] in Table 3.8.

3.4.5 Conclusions

The results demonstrated above show that the LBFS can calculate the steady state solution of the lid-driven cavity problem to an acceptable level of accuracy. However the predictions are still not as accurate as the results presented by Wang

Re	Vortex Centre (x,y)		
	N_x	Ghia et al.[4]	LBFS
100	50	(0.6172,0.7344)	(0.6213,0.7568)
100	129	(0.6172,0.7344)	(0.6208,0.7460)
400	129	(0.5547,0.6055)	(0.5641,0.6153)
1000	129	(0.5313,0.5625)	(0.5357,0.5721)
3200	129	(0.5165,0.5469)	(0.5185,0.5450)

TABLE 3.8: Co-ordinates of primary vortex centres for individual test case.

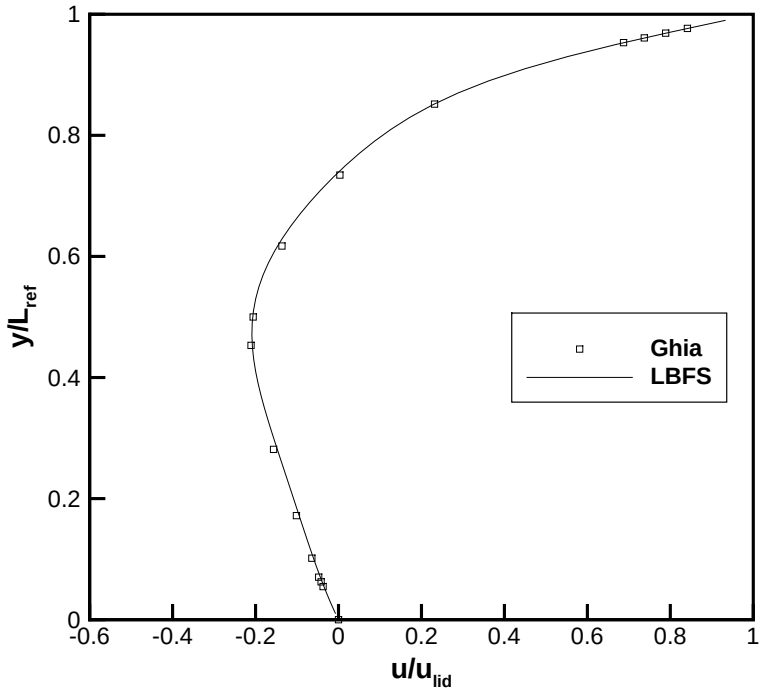
et al. [136]. This may be due to the differences between the approaches taken and these include:

- Use of non-uniform grids by Wang et al.
- Direct specification of boundary conditions by Wang et al.

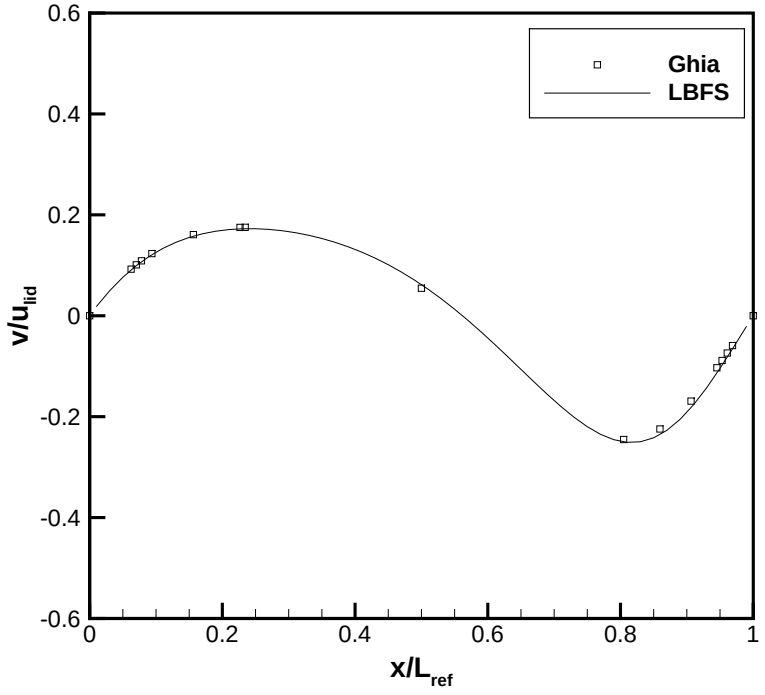
The main difference is that Wang et al. used a non-uniform grid which was more refined at the boundary and had larger cells in the interior. This would aid in resolving the complex flow conditions at the boundary of the cavity.

Wang et al. also directly calculated the boundary fluxes from the N-S equations. This could eliminate any source of errors in the flux calculation that could be related to the use of the ghost cell method in this work (see Section 2.2.3).

To conclude this flow problem has verified that the LBFS can predict complex 2D flows for a variety of Reynolds numbers.

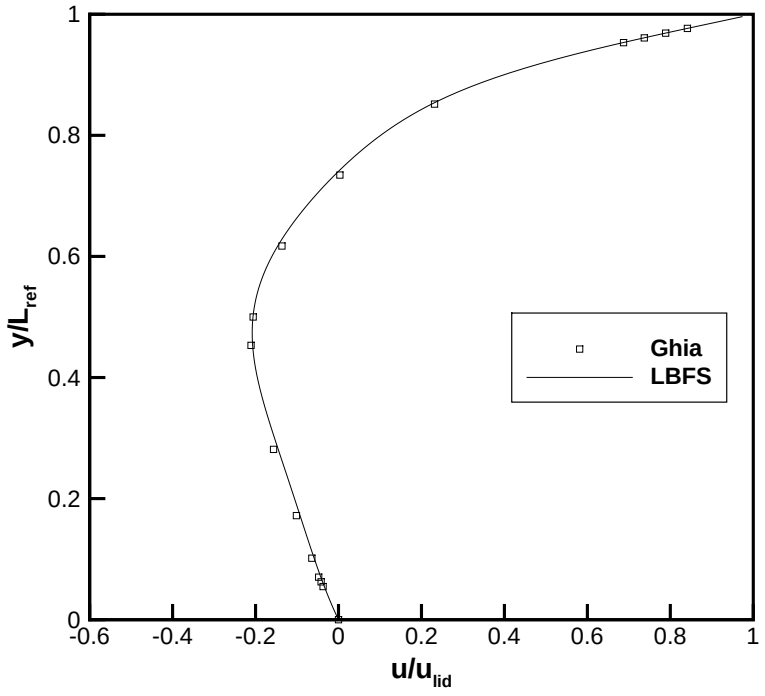


(a)

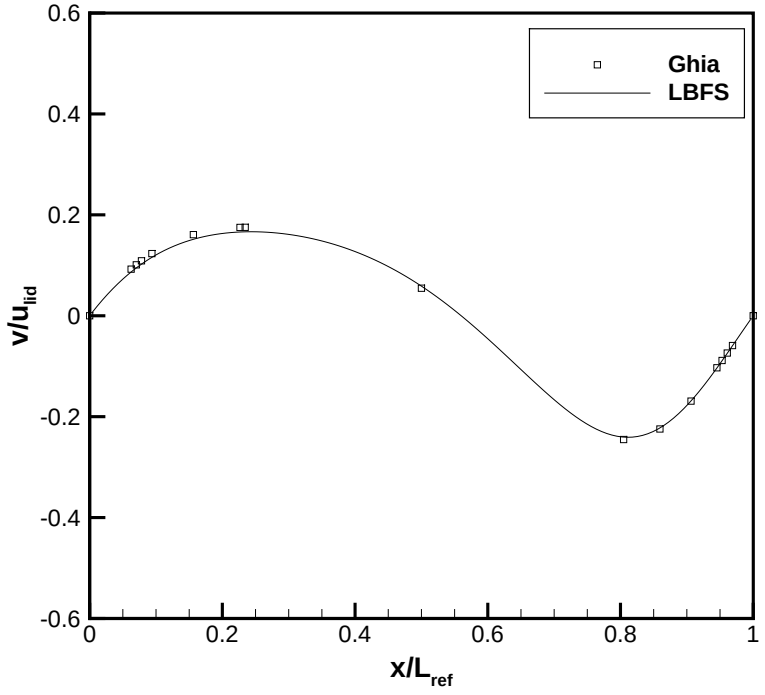


(b)

FIGURE 3.7: 2D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 100$ and $N_x = 50$.

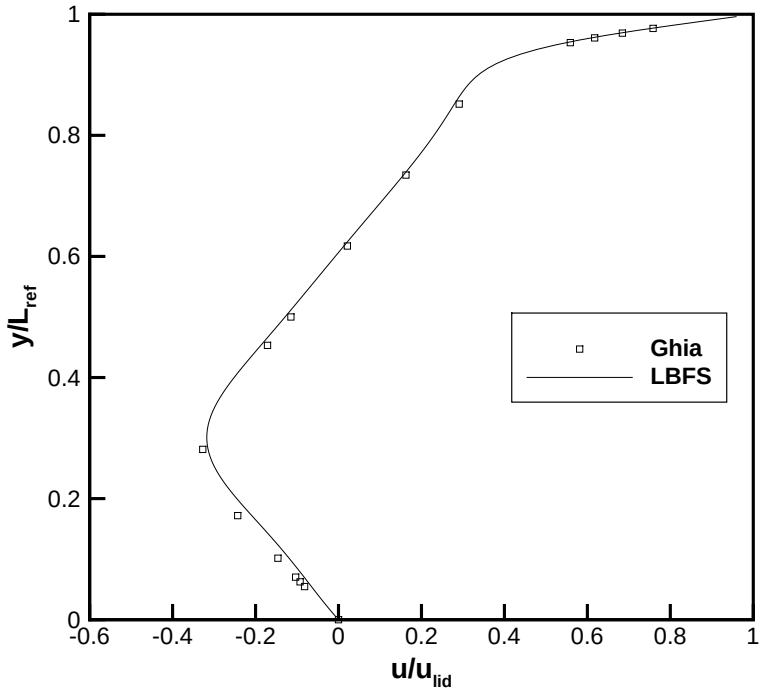


(a)

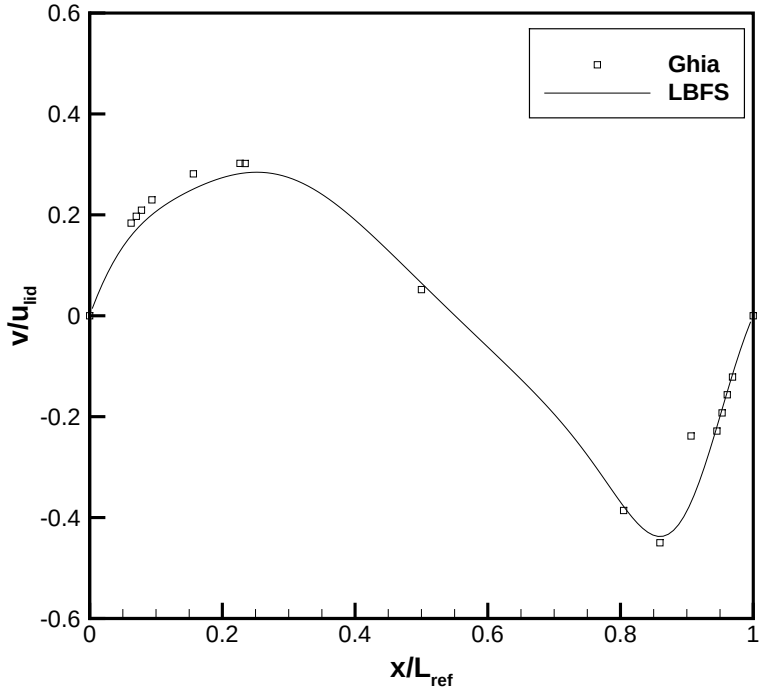


(b)

FIGURE 3.8: 2D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 100$ and $N_x = 129$.

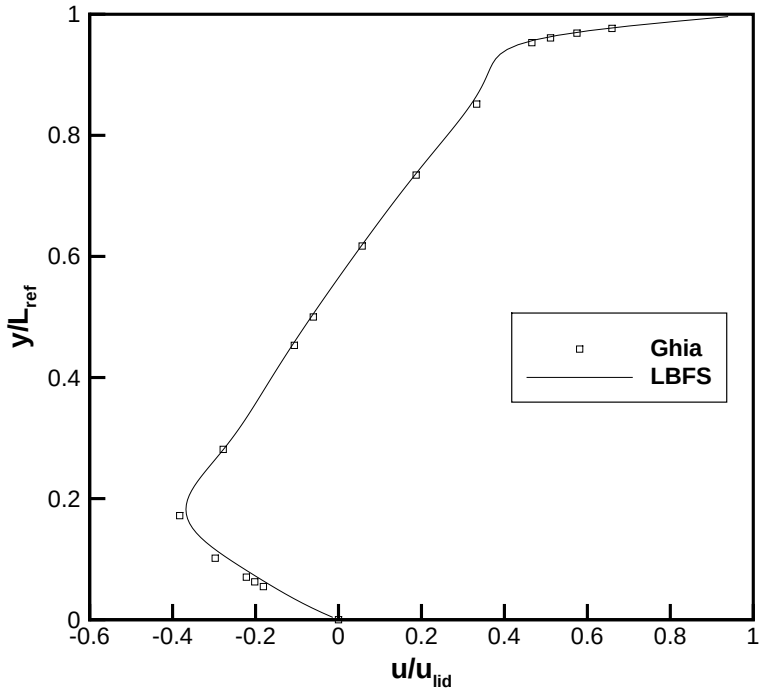


(a)

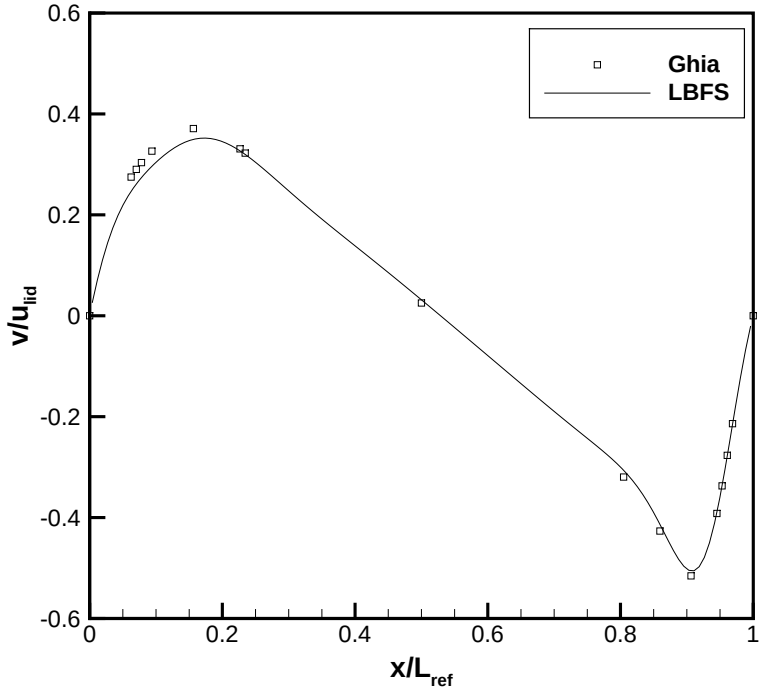


(b)

FIGURE 3.9: 2D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 400$ and $N_x = 129$.

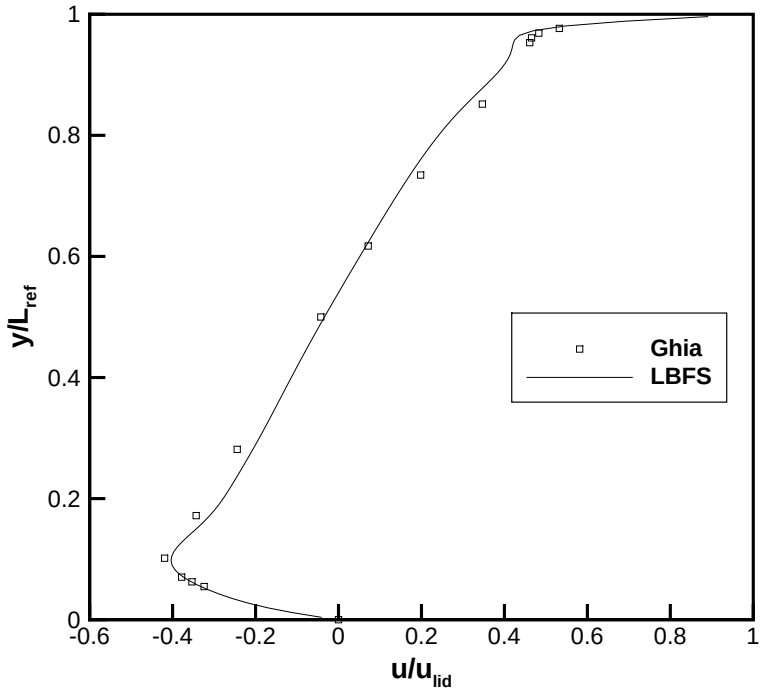


(a)

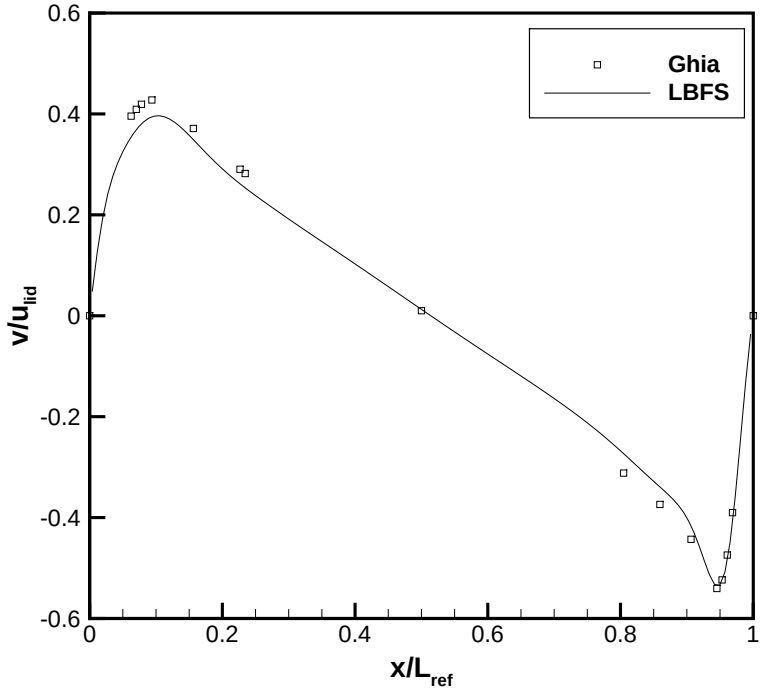


(b)

FIGURE 3.10: 2D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 1000$ and $N_x = 129$.

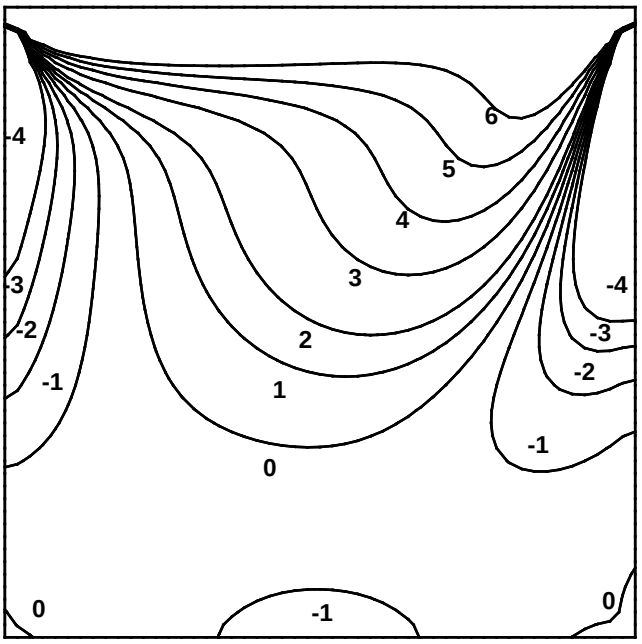


(a)

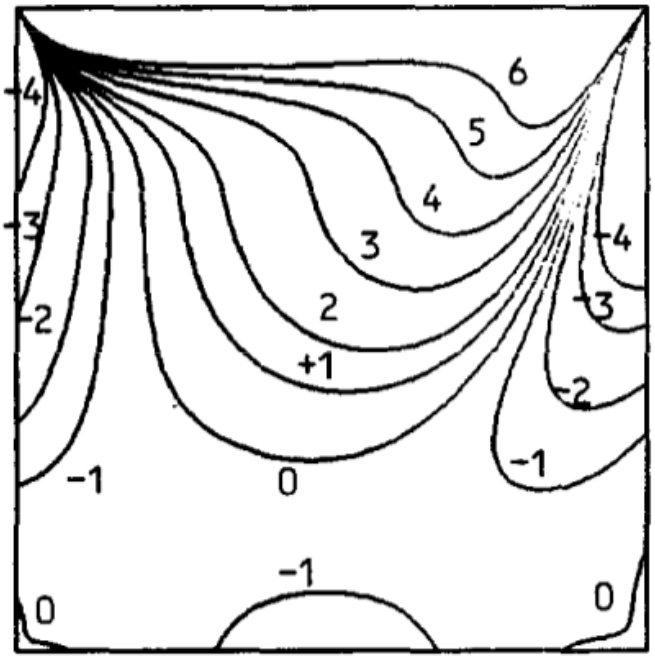


(b)

FIGURE 3.11: 2D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 3200$ and $N_x = 129$.

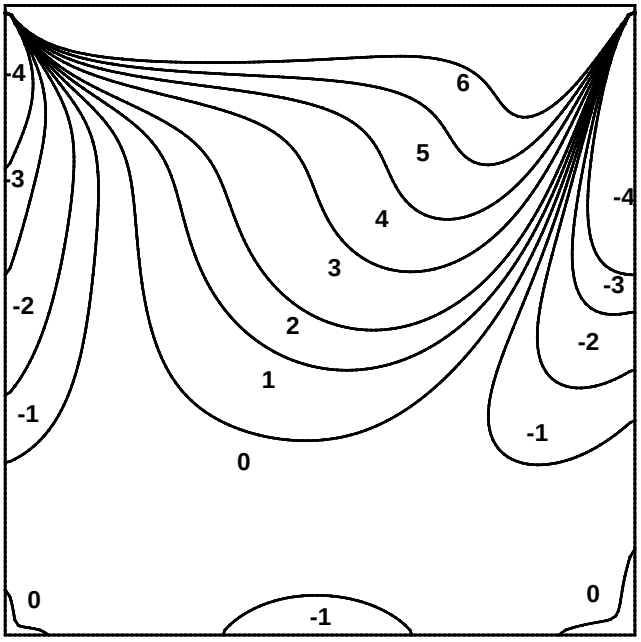


(a)

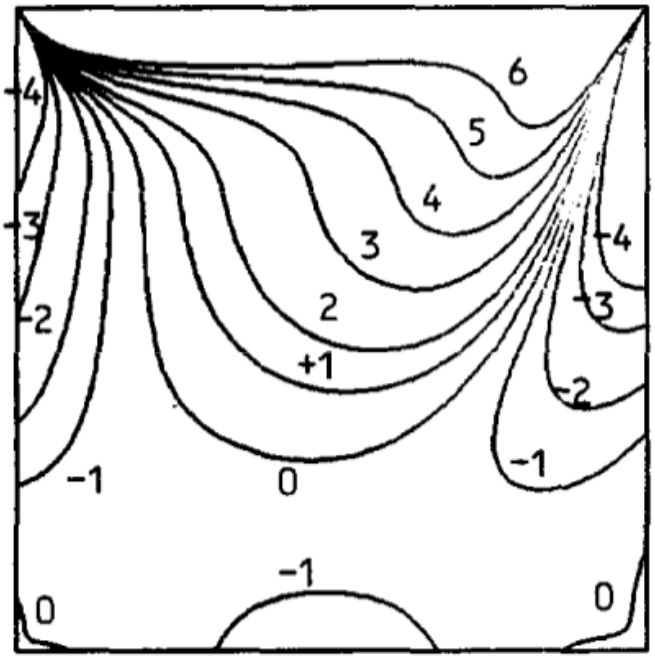


(b)

FIGURE 3.12: 2D lid-driven cavity flow: z vorticity plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 100$ and $N_x = 50$.

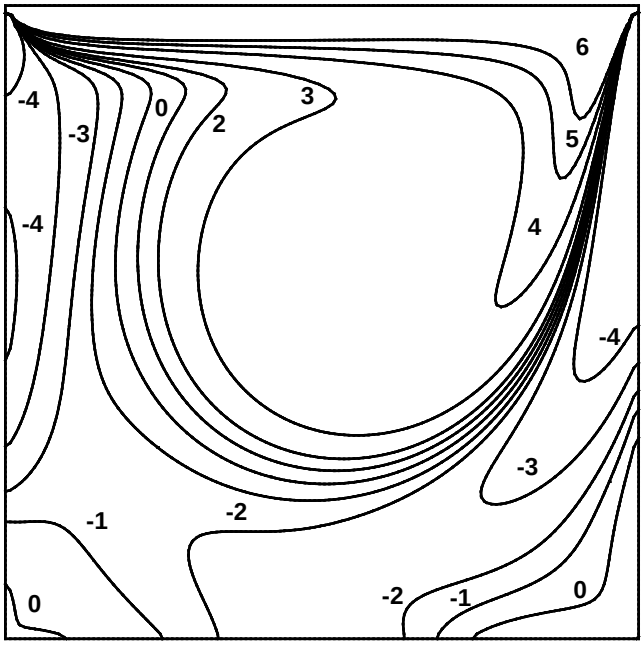


(a)

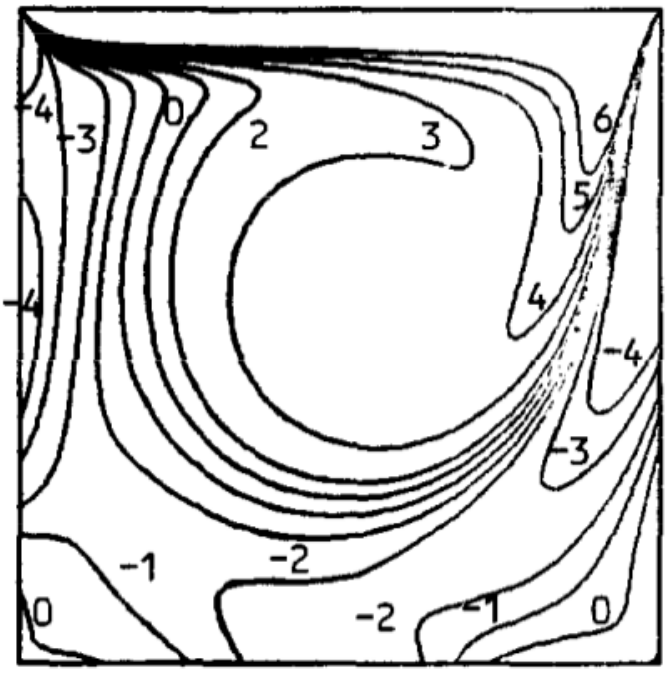


(b)

FIGURE 3.13: 2D lid-driven cavity flow: z vorticity plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 100$ and $N_x = 129$.

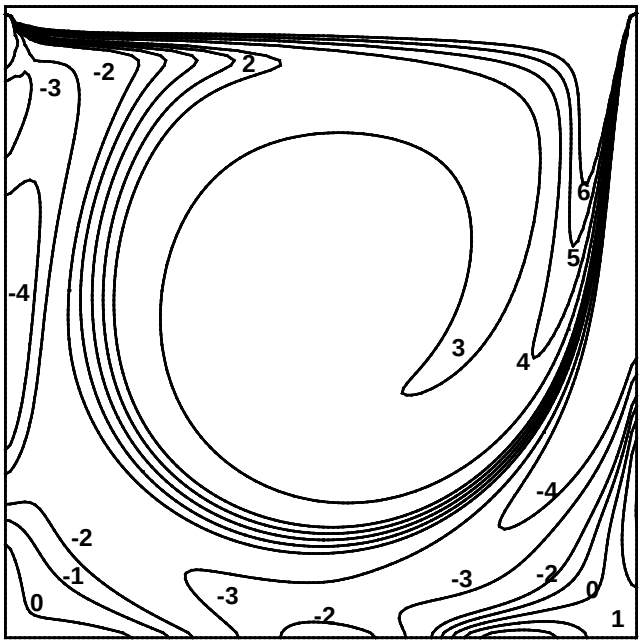


(a)

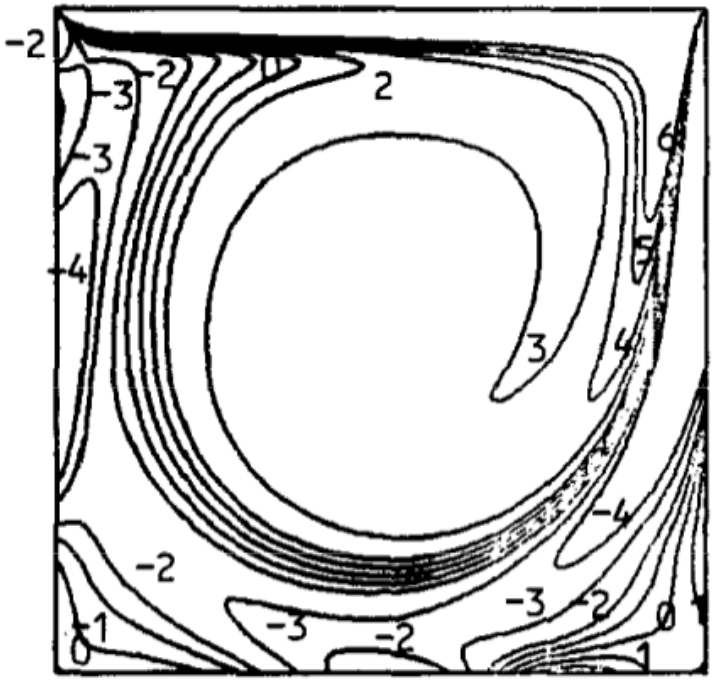


(b)

FIGURE 3.14: 2D lid-driven cavity flow: z vorticity plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 400$ and $N_x = 129$.

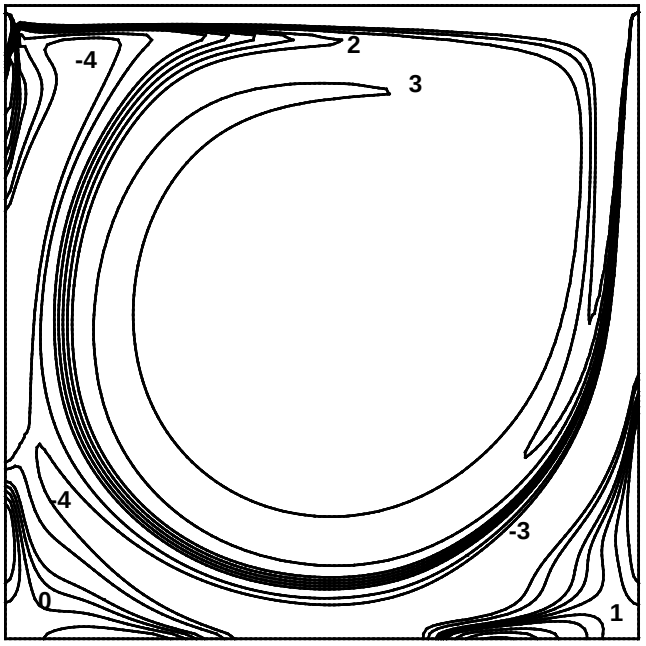


(a)

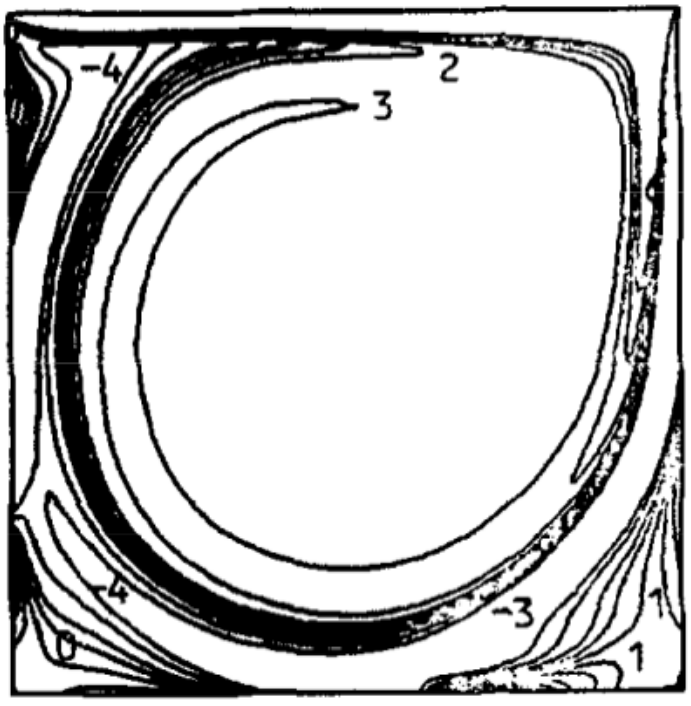


(b)

FIGURE 3.15: 2D lid-driven cavity flow: z vorticity plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 1000$ and $N_x = 129$.

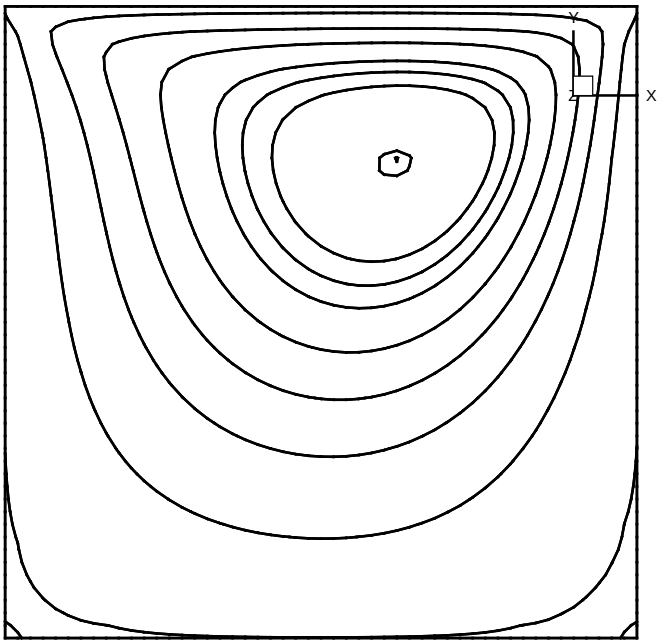


(a)

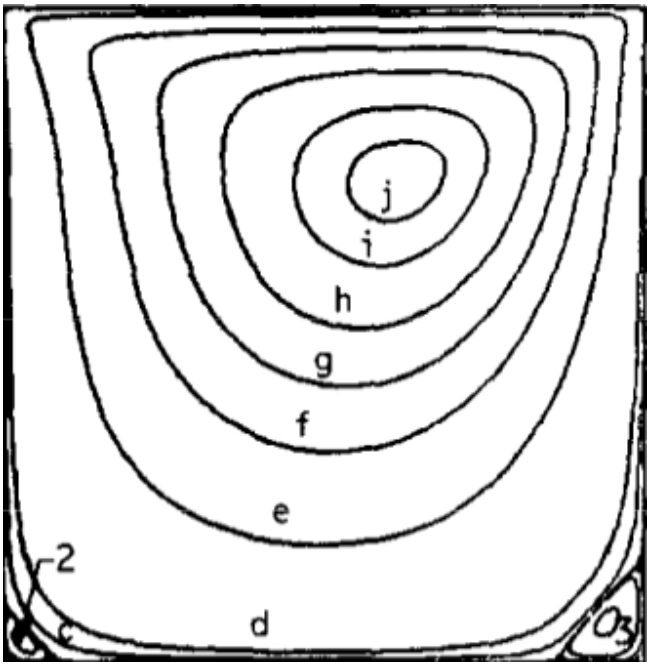


(b)

FIGURE 3.16: 2D lid-driven cavity flow: z vorticity plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 3200$ and $N_x = 129$.

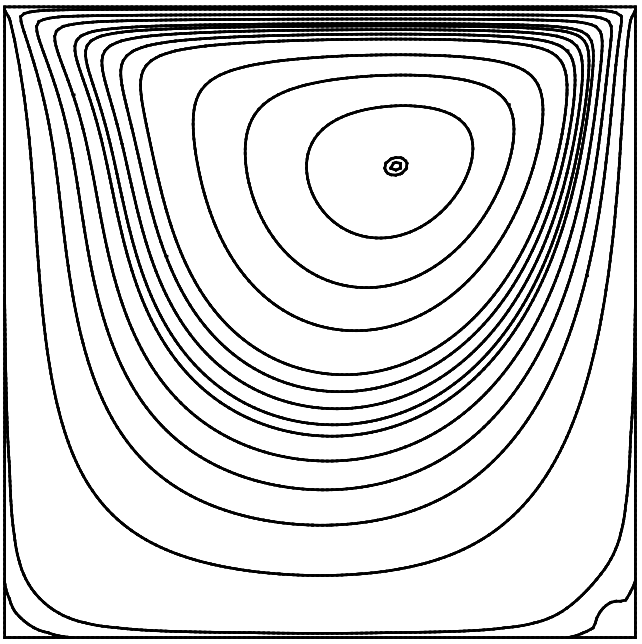


(a)

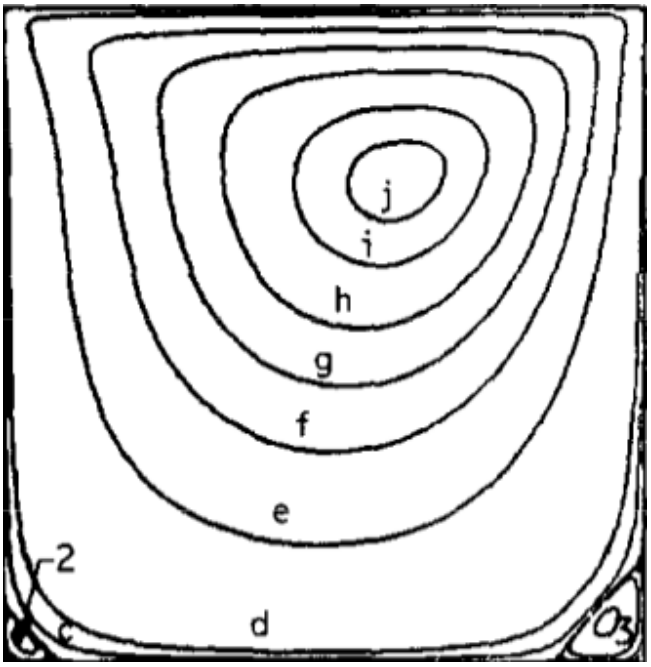


(b)

FIGURE 3.17: 2D lid-driven cavity flow: streamline plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 100$ and $N_x = 50$.

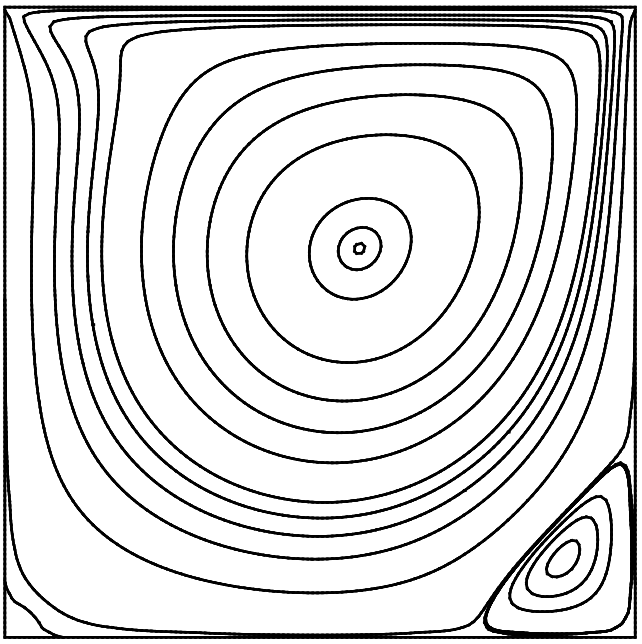


(a)

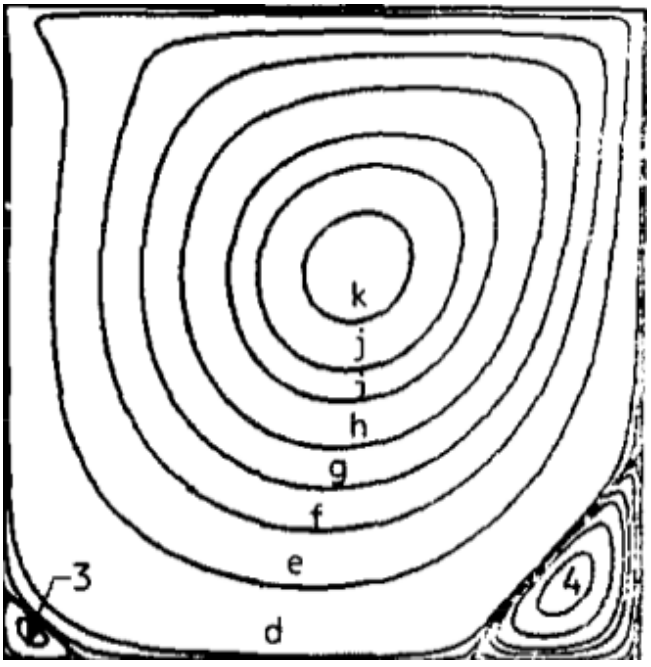


(b)

FIGURE 3.18: 2D lid-driven cavity flow: streamline plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 100$ and $N_x = 129$.

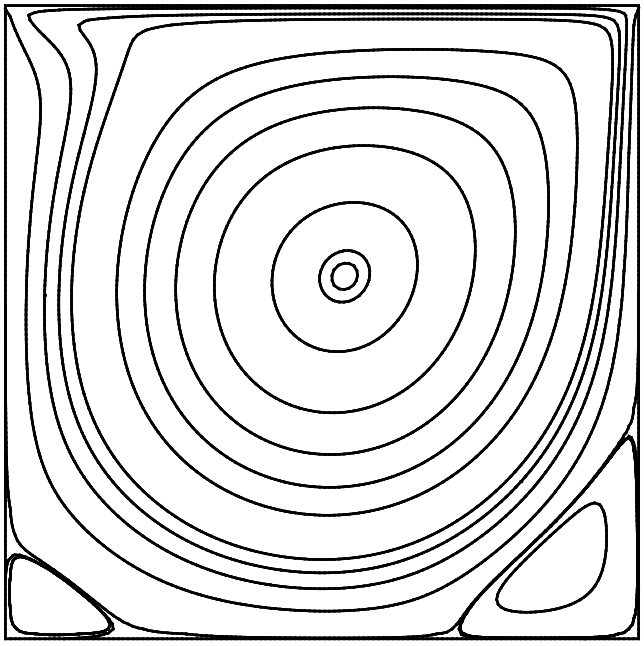


(a)



(b)

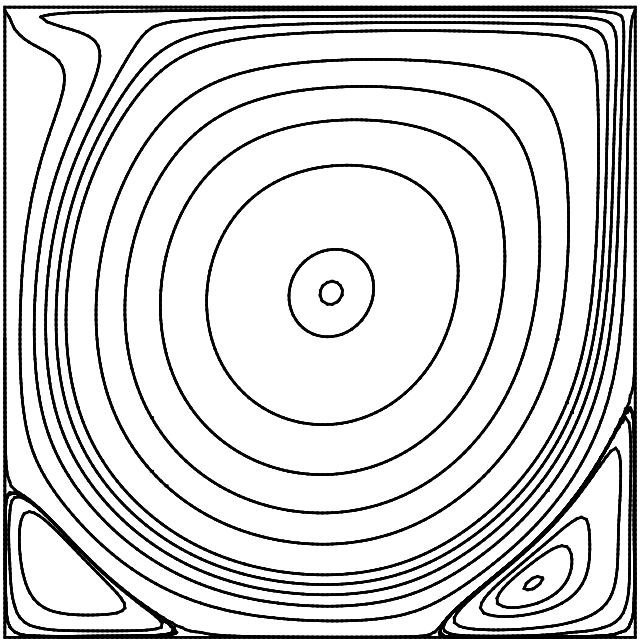
FIGURE 3.19: 2D lid-driven cavity flow: streamline plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 400$ and $N_x = 129$.



(a)

(b)

FIGURE 3.20: 2D lid-driven cavity flow: streamline plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 1000$ and $N_x = 129$.



(a)

(b)

FIGURE 3.21: 2D lid-driven cavity flow: streamline plots for a) LBFS and b) Ghia[4] on the $z = 0$ plane for $Re = 3200$ and $N_x = 129$.

3.5 Taylor-Green Vortex Flow

3.5.1 Introduction

Taylor-Green vortex flow is a transient flow problem in which a vortex is set as an initial condition and the vortex subsequently decays exponentially with time. It has an analytical solution and this can be used to test the spatial and temporal accuracy of N-S solvers.

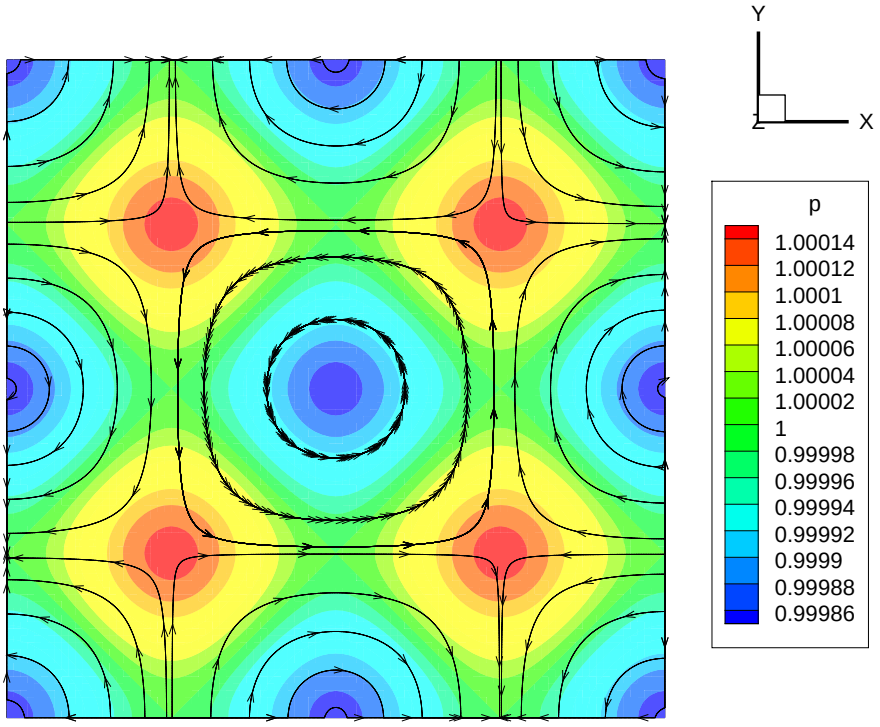


FIGURE 3.22: Taylor-Green vortex flow - initial conditions for pressure and initial streamlines.

3.5.2 Problem Set Up

Taylor-Green flow is a transient flow problem and is fully periodic in a computational domain of L_x x L_y. In 2D space the velocity and pressure fields can be

calculated using Equation (3.15) and Equation (3.16) respectively:

$$\mathbf{V}(\mathbf{r},t) = u_0 \begin{pmatrix} -\sqrt{\frac{k_y}{k_x}}\cos(k_x x)\sin(k_y y) \\ -\sqrt{\frac{k_x}{k_y}}\sin(k_x x)\cos(k_y y) \end{pmatrix} e^{-t/t_d} \tag{3.15}$$

and

$$p(\mathbf{r},t) = p_0 - \rho\frac{u_0^2}{4}\left(\frac{k_y}{k_x}\cos(2k_x x) + \frac{k_x}{k_y}\cos(2k_y y)\right)e^{-2t/t_d} \tag{3.16}$$

where u_0 is the initial velocity scale, $k_{x,y} = \dfrac{2\pi}{L_{x,y}}$ are the components of the wave vector $\mathbf{k}$ and

$$t_d = \frac{1}{\nu\left(k_x^2 + k_y^2\right)} \tag{3.17}$$

is the vortex decay time. The average pressure p_0 is an arbitrary value and does not enter the N-S equations.

In the test cases performed L_x was set equal to L_y. This resulted in the following velocity and pressure fields:

$$\mathbf{V}(\mathbf{r},t) = u_0 \begin{pmatrix} -\cos\left(\frac{2\pi}{L_{x,y}}x\right)\sin\left(\frac{2\pi}{L_{x,y}}y\right) \\ -\sin\left(\frac{2\pi}{L_{x,y}}x\right)\cos\left(\frac{2\pi}{L_{x,y}}y\right) \end{pmatrix} e^{-t/t_d} \tag{3.18}$$

and

$$p(\mathbf{r},t) = p_0 - \rho\frac{u_0^2}{4}\left(\cos\left(\frac{4\pi}{L_{x,y}}x\right) + \cos\left(\frac{4\pi}{L_{x,y}}y\right)\right)e^{-2t/t_d} \tag{3.19}$$

where the decay time is given by:

$$t_d = \frac{L_{x,y}^2}{8\pi^2\nu} \tag{3.20}$$

Periodic boundary conditions were implemented at all boundaries. The liquid in the flow was chosen to be water. A Reynolds number of 10 was also chosen. The test case parameters used are detailed in SI units in Table 3.9.

3.5.3 Results

There are many objectives to the running of this benchmark flow problem. These include:

Variable Name	Value	Units
$L_x = L_y$	0.1	m
ν	$1.04 * 10^{-6}$	$\mathrm{Pa\,s^{-1}}$
u_0	$1.04 * 10^{-4}$	$\mathrm{m\,s^{-1}}$
ρ	1020.5	$\mathrm{kg\,m^{-3}}$
Reynolds No.	10	-
p_0	$2.54 * 10^{-6}$	Pa

TABLE 3.9: Overall test case parameters for decaying vortex flow.

1. Comparison of numerical solution of velocities to analytical solution at centerline.
2. Comparison with existing LBFS results in the literature of second-order accuracy in space.
3. Comparison of LBFS with LBM in both accuracy and order of accuracy.
4. Investigation of accuracy with varying Mach number.

To compare the claim of second-order accuracy in space for the LBFS, the test cases in Table 3.10 were run. This involved running the test case at a variety of mesh densities (where $N_{x,y}$ is the number of cells in x and y direction). The diffusive scaling approach was used in the non-dimensionalisation of the parameters. These are the exact same values as those used by Wang et al. [136] whose results will be used as a comparison.

Test Case No.	Mesh Density (No. of Cells)	$N_{x,y}$	$\frac{\Delta x}{\delta x}$	$\Delta t(s)$
1	441	21	2	0.2857
2	1681	41	2	0.0731
3	6561	81	2	0.0185
4	25921	161	2	0.0046

TABLE 3.10: Individual test case parameters varying with mesh density for decaying vortex flow for comparison with existing LBFS results.

Shardt [185] has published a Matlab code that demonstrates that the LBM is second-order accurate in space. The LBFS was run for these same test cases. These test cases are detailed in Table 3.11.

To investigate the temporal order of accuracy the test case was run for a variety of time steps for the mesh density where $N_{x,y} = 96$,. These are summarised in Table 3.12.

Test Case No.	Mesh Density (No. of Cells)	$N_{x,y}$	$\frac{\Delta x}{\delta x}$	$\Delta t(s)$
5	256	16	2	0.3125
6	1024	32	2	0.0781
7	4096	64	2	0.00195
8	9216	96	2	0.00086

TABLE 3.11: Individual test case parameters varying with mesh density for decaying vortex flow for comparison with LBM.

Test Case No.	Mesh Density (No. of Cells)	$N_{x,y}$	$\frac{\Delta x}{\delta x}$	$\Delta t(s)$
9	9216	96	2	0.00862
10	9216	96	2	0.00431
11	9216	96	2	0.00215
12	9216	96	2	0.00107

TABLE 3.12: Individual test case parameters varying with time step for $N_{x,y} = 96$ for decaying vortex flow.

To measure the accuracy of the LBFS, the error in the numerical solution versus the analytical solution was calculated using a L_2 norm of the error in the velocity. For every test case the error was calculated at the time $\frac{t}{t_d} = 1$. Wang et al. [136] and Shardt [185] use variations of the L_2 norm to calculate the error. These were used where appropriate to ensure a like-with-like comparison. Wang et al. [136] use the volume averaged L_2 norm which is given by Equation (3.21):

$$L_2\left(u\right)_{relative} = \sqrt{\frac{1}{N_x \times N_y} \sum_{1}^{No\ of\ Cells} \left[\frac{u^{numerical} - u^{analytical}}{u^{analytical}}\right]^2} \tag{3.21}$$

Shardt uses a L_2 norm of the total error to measure the error. This is given in Equation (3.22):

$$L_2\left(u\right)_{total} = \sqrt{\sum_{1}^{No\ of\ Cells} \left[\frac{u^{numerical} - u^{analytical}}{u^{analytical}}\right]^2} \tag{3.22}$$

The L_2 norm at time $\frac{t}{t_d} = 1$ was plotted against mesh density and time step to investigate the spatial and temporal orders of accuracy.

The velocity profiles of the vortex solution at the decay time were calculated across the centrelines of the domain in the x and y direction. The velocity profiles were then compared to the analytical solution at three different times $\frac{t}{t_d} = 0.2038$,

$\frac{t}{t_d} = 0.6115$ and $\frac{t}{t_d} = 1$ The results for Test Case 8 are shown in Figures 3.23 to 3.24.

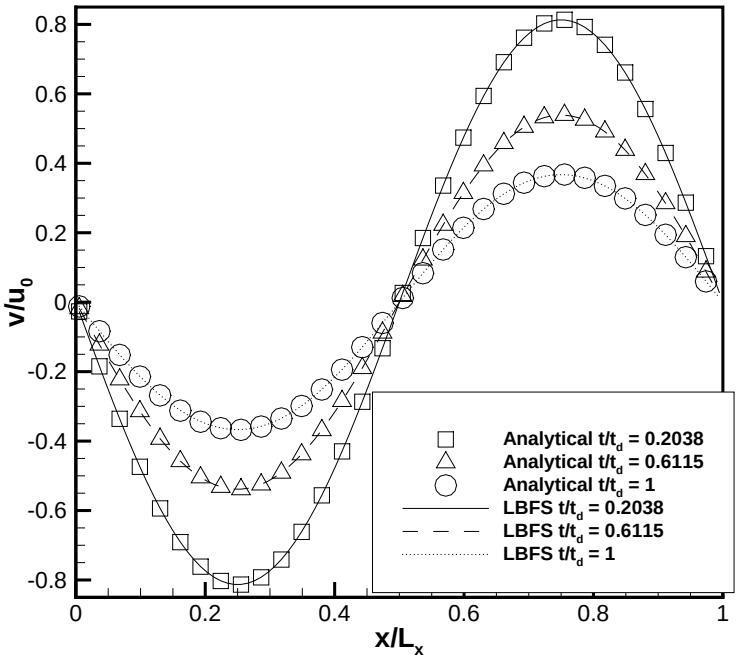


FIGURE 3.23: LBFS $Re = 10, N_x = 96$ VX velocity profile at Y=0.5 for Taylor-Green vortex flow at varying times.

To investigate the spatial order of accuracy of the LBFS, Test Cases 1-4 were run (see Table 3.10). The L_2 norm of the error was calculated for each test case at the time $\frac{t}{t_d} = 1$ using Equation (3.21). The results are plotted in Figure 3.25. A linear curve was fitted to the data using Tecplot (Bellevue, Washington, United States) and it resulted in a linear curve with a slope value of 2.0543. This indicates that the LBFS is approximately second-order accurate in space. This agrees with the results of Wang et al. [136] who provided a result of 1.971. To investigate the accuracy of the LBFS in comparison with the LBM, Test Cases $5 - 8$ were run (see Table 3.11). The L_2 norm of the error was calculated for each test case at the time $\frac{t}{t_d} = 1$ using Equation (3.21). This is the metric that Wang et al. [136] used and the results can be seen in Figure 3.26. This figure shows that the LBM is third-order accurate in space. However the LBM is formally described as being second-order in space [186]. Shardt's LBM code [185] is documented as showing the LBM has second-order spatial accuracy. This contradicts the results shown in Figure 3.26. To investigate this contradiction the LBFS and LBM code were run

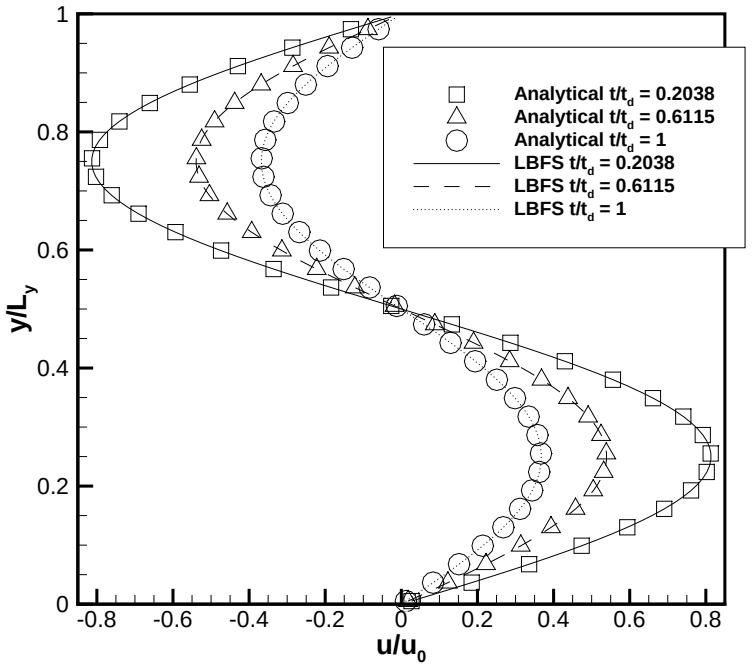


FIGURE 3.24: LBFS $Re = 10, N_x = 96$ UY velocity profile at Y=0.5 for Taylor-Green vortex flow at varying times.

for Test Cases $5-8$ (see Table 3.11) using Shardt's measure of L_2 norm shown in Equation (3.22). These results are shown in Figure 3.27. This results in the LBFS having a spatial order of accuracy of approximately 3.3 for this set of test cases. For these test cases the LBFS also outperforms the LBM code in both absolute accuracy and also the spatial order of accuracy. The spatial order of accuracy of the LBFS is shown to improve for Test Cases $5-8$. The big difference between these cases and Test Cases $1-4$ is the initial Mach number chosen for diffusive scaling. Cases $1-4$ have a starting Mach number of 0.206 for a finer mesh, whereas cases $5-8$ have a starting Mach number of 0.017 for a coarser mesh. The effect of Mach number on acccuracy can be seen in Figure 3.28.

To ascertain the temporal order of accuracy, the numerical experiment was run for a variety of time steps for the mesh density where $N_{x,y} = 96$. These test cases are summarised in Table 3.12. The results can be seen in Figure 3.29. It can be seen that the LBFS is approximately fourth order accurate in time. As a RK4 scheme is used and is theoretically fourth order accurate, this result shows that the RK4 scheme is correctly implemented.

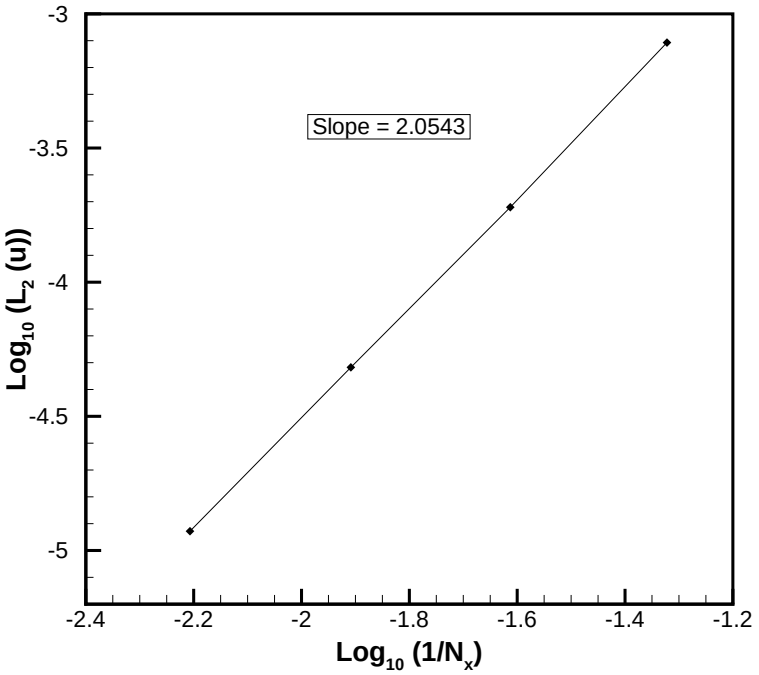


FIGURE 3.25: LBFS - volume averaged L_2 norm error of numerical solution vrs mesh density for Re=10 Taylor-Green vortex flow.

3.5.4 Conclusions

Firstly the results of Wang et al. [136] were recreated and it was shown that the LBFS is at least second order accurate in space when using the volume averaged L_2 norm error metric. However when this approach was used to compare the LBFS to the LBM, it can be seen that the volume averaged L_2 norm error metric was not a suitable metric as it resulted in making the LBM third order in space. It is noted in the literature that the LBM is second-order accurate in space [186]. Hence the total L_2 norm error metric would appear to be a more accurate reflection of the spatial order of accuracy.

When using the total L_2 norm error metric, it appears that the LBFS can perform somewhere between first and third order accurate in space depending on the Mach number. This increase in accuracy is due to compressibility errors generated by the lack of third order term in the equilibrium distribution function. As this error is of the order $O(V^3)$, it would be expected to see large decreases in error as a result of a reduction in Mach number. Increasing mesh density would have little impact in reducing this error.

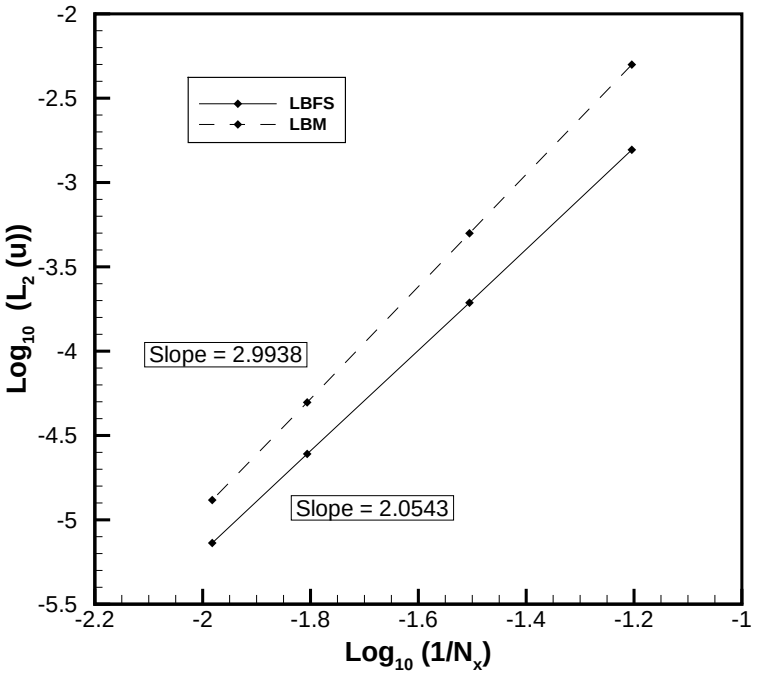


FIGURE 3.26: LBFS and LBM comparison - volume averaged L_2 norm error of numerical solution vrs mesh density for Re=10 Taylor-Vortex flow.

This benchmark flow problem also shows that the LBFS can produce time accurate simulations for transient flow problems. As shown in Figures 3.23 to 3.24 , there is excellent agreement between the predicted velocity profiles and the analytical solution at a variety of times. Finally the temporal order of accuracy is found to be 4.05. The RK4 method is theoretically fourth order accurate and this shows that the RK4 method is correctly implemented.

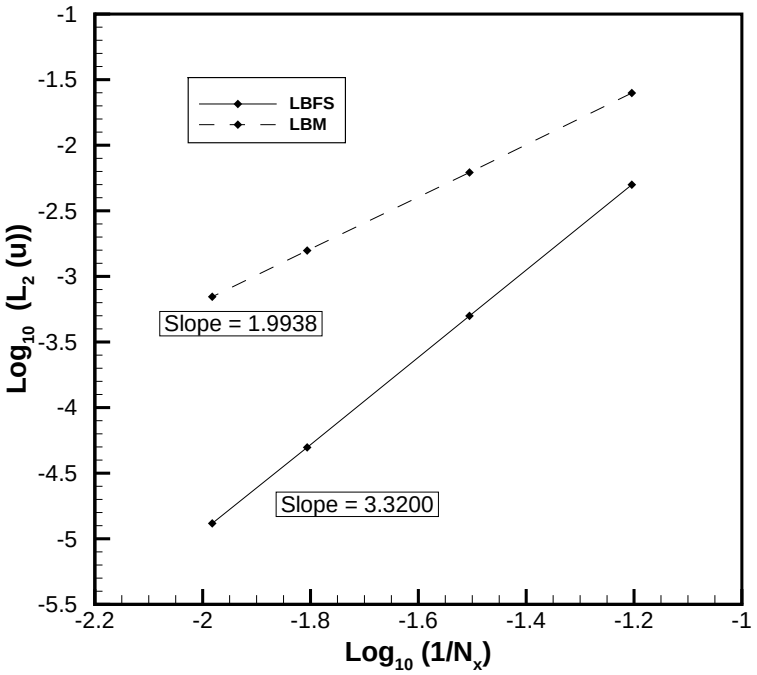


FIGURE 3.27: LBFS and LBM comparison - total L_2 norm error of numerical solution vrs mesh density for Re=10 Taylor-Green vortex flow.

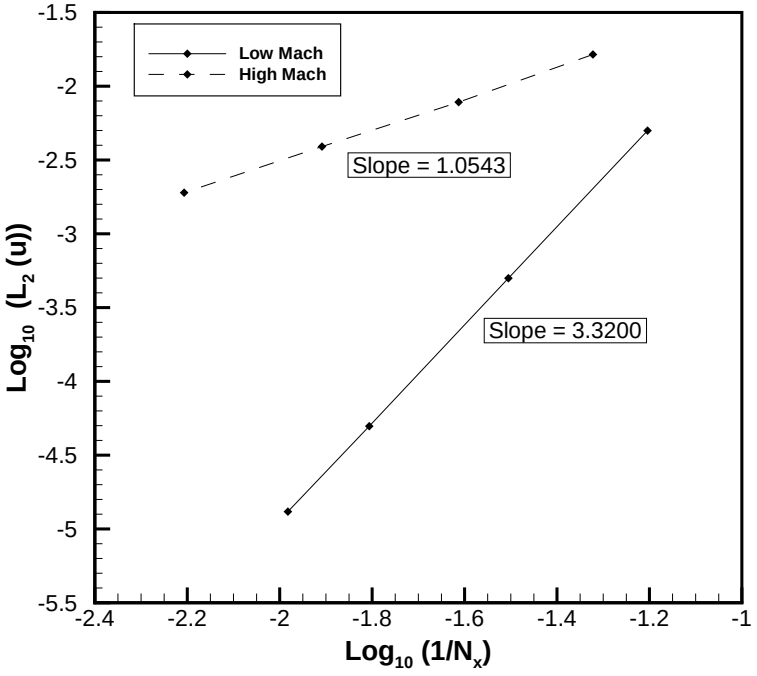


FIGURE 3.28: LBFS high Mach no. and low Mach no. comparison - total L_2 norm error of numerical solution vrs mesh density for Re=10 Taylor-Green vortex flow.

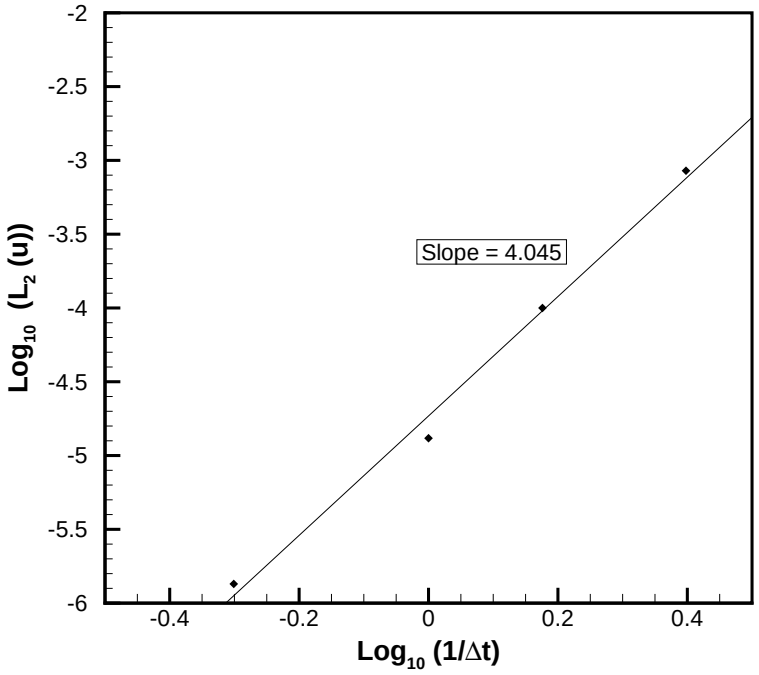


FIGURE 3.29: LBFS - total L_2 norm error of numerical solution vrs time step for Re=10 Taylor-Green vortex flow.

3.6 Womersley Flow

3.6.1 Introduction

Womersley flow is a transient flow problem in which a pressure gradient which varies periodically in time is applied along a channel. It has as an analytical solution and this can be used to test the spatial and temporal accuracy of N-S solvers for unsteady flow.

A dimensionless quantity which serves as a general-purpose indicator of the nature of the unsteady flow is the Womersley number Wo. This is given by:

$$Wo = \frac{h}{2}\sqrt{\frac{\omega}{\nu}} \tag{3.23}$$

where h is the channel height, ω is the angular frequency of the unsteady flow equal to $2\pi/T$, ν is the kinematic viscosity and T is the period of the oscillations in the flow.

3.6.2 Problem Set Up

The analytical solution of Womersley flow is for static plates of finite length and a periodically varying pressure differential between the inlet and outlet. This was implemented in the computational domain by implementing a no slip boundary condition for velocity at $y = -a$ and $y = a$ ($a = h/2$). The pressure differential was implemented by using a body force whose magnitude varies periodically in time (see Figure 3.30).

An analytical solution can be derived which gives differing velocity profiles for differing Womersley numbers and times within the flow period. The influence of time and Womersley number can be seen in Figure 3.31.

The analytical solution for the axial velocity u is given in complex form as:

$$u\left(y,t\right) = \frac{A}{i\omega\rho}\left[1 - \frac{\cosh\left(Wo\ i^{0.5}\ \dfrac{y}{a}\right)}{\cosh\left(Wo\ i^{0.5}\right)}\right] e^{i\omega t} \tag{3.24}$$

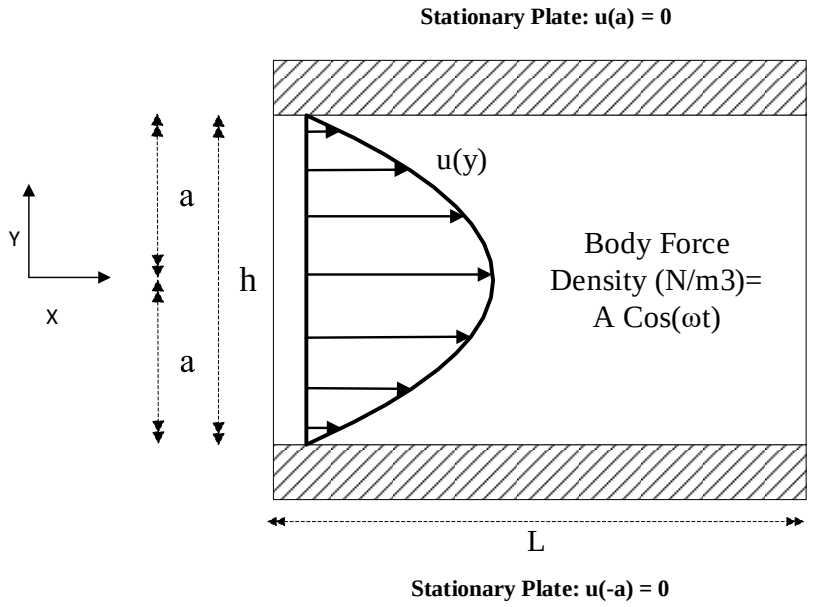


FIGURE 3.30: Womersley flow - set up with example parabolic profile for Wo=1.

where A is the amplitude of the body force density driving the flow. The real part of Equation (3.24) gives the explicit solution of velocity for Womersley flow. As shown by [5] this is given by:

$$u\left(y,t\right)=\frac{A}{\omega\rho\gamma}\left\{\begin{array}{c}\left[\sinh\phi_1(y)sin\phi_2(y)+\sinh\phi_2(y)sin\phi_1(y)\right]\cos(\omega t)\\+\left[\gamma-\cosh\phi_1(y)cos\phi_2(y)+\cosh\phi_2(y)cos\phi_1(y)\right]\sin(\omega t)\end{array}\right\} \tag{3.25}$$

where the pressure gradient varies in time as follows:

$$\frac{\partial p\left(t\right)}{\partial x}=-A\cos\left(\omega t\right) \tag{3.26}$$

A is calculated from the following formula:

$$A=\frac{\mu^2 Re}{\rho a^3} \tag{3.27}$$

A Reynolds number of 250 was chosen so as to reduce the period of the pressure oscillations and reduce runtimes. A Womersley number of 10 was chosen as this results in the most complex velocity profiles. h, μ and ρ were chosen to reflect typical values found within aortic blood flow. This results in a characteristic velocity (u_0) of the flow of $0.07232\mathrm{m\,s^{-1}}$. The test case parameters used in this

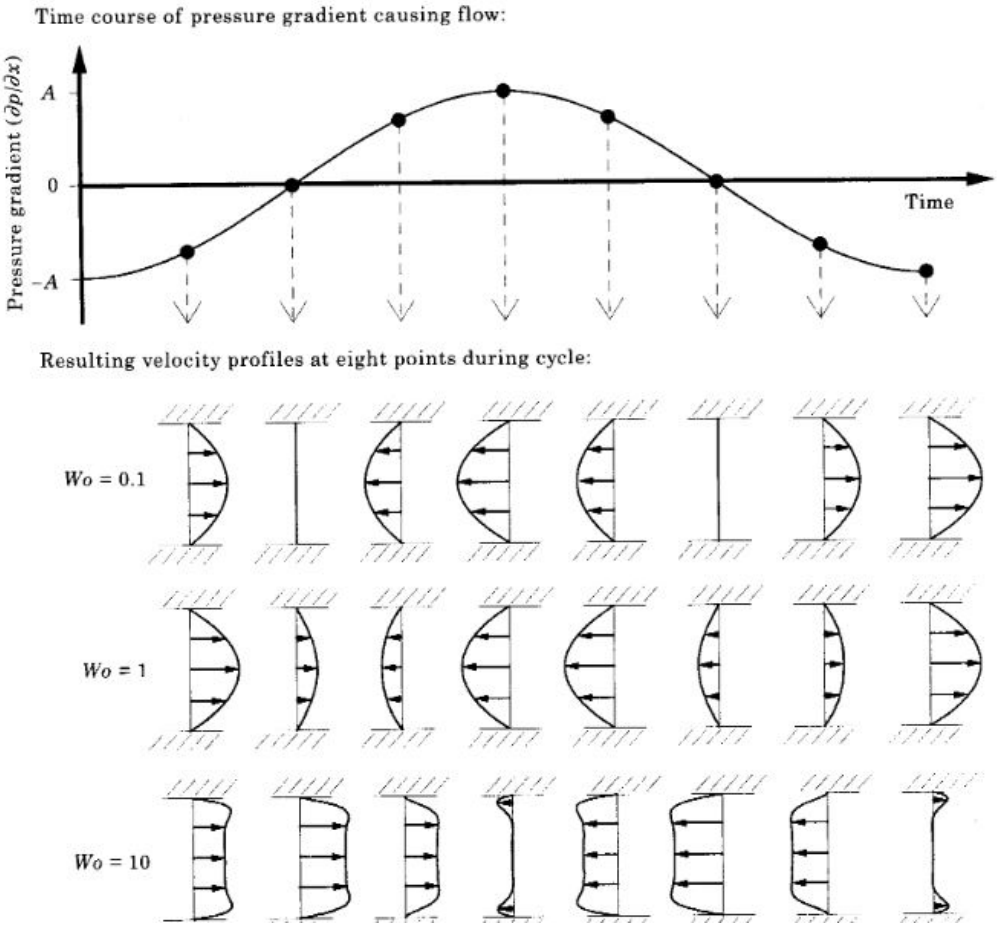


FIGURE 3.31: Velocity profiles between two flat plates at eight points in time during a single cycle of a sinusoidally-varying pressure gradient for three values of Wo. [5]

test case are detailed in SI units in Table 3.13. A sensitivity analysis was performed

Variable Name	Value	Units
h	0.0043	m
μ	0.001275	Pa s
ρ	1020.5	$\mathrm{kg\,m^{-3}}$
Reynolds No.	250	-
A	39.895	$\mathrm{N\,m^{-3}}$
u_0	0.07232	$\mathrm{m\,s^{-1}}$
Womersley No.	10	-

TABLE 3.13: Overall test case parameters for Womersley flow.

on the length of the channel and it was found that various channel lengths resulted in the same results. As a result it was decided to have a channel length of five cells in the x direction. This reduced the computational time required to run the flow problem.

3.6.3 Results

3.6.3.1 Introduction

The objectives behind running this benchmark problem include:

1. Comparison of the predicted solution of the axial velocity across channel to analytical solution at a particular time and Womersley number.
2. Comparison of the predicted solution of the centerline axial velocity of channel to analytical solution at a particular time for a given Womersley number.

A mesh sensitivity analysis was performed with a variety of mesh densities listed in Table 3.14. Mesh convergence was achieved for Test Case 4 and all results are shown for this test case. All test cases were run for a Womersley number of 10 as this produces the most complex velocity profiles.

Test Case No.	Mesh Density (No. of Cells)	N_y	N_x	$\frac{\Delta x}{\delta x}$	Ma	$\frac{T}{\Delta t}$
1	50	10	5	2	1.08253	126
2	100	20	5	2	0.54126	314
3	200	40	5	2	0.27063	1256
4	250	50	5	2	0.216	3142

TABLE 3.14: Individual test case parameters used in Womersley flow.

The u-velocity profiles were predicted across the height of the channel for Test Case 4 at each time step. The velocity profiles were then compared to the analytical solution at different times in the third cycle of oscillations of the flow. These are $3.125T$, $3.25T$, $3.375T$, $3.50T$, $3.625T$, $3.75T$, $3.875T$ and $4.00T$.

The results for Test Case 4 are shown in Figure 3.32 and Figure 3.33.

The predicted u-velocity at $y = 0$ for Test Case 4 was plotted against time. It is compared to the analytical solution and also to the driving force. The results are shown in Figure 3.34.

3.6.4 Conclusions

Results of the Womersley flow problem was predicted using the LBFS for a variety of mesh densities. Solutions were predicted for a Womersley number of 10 as this

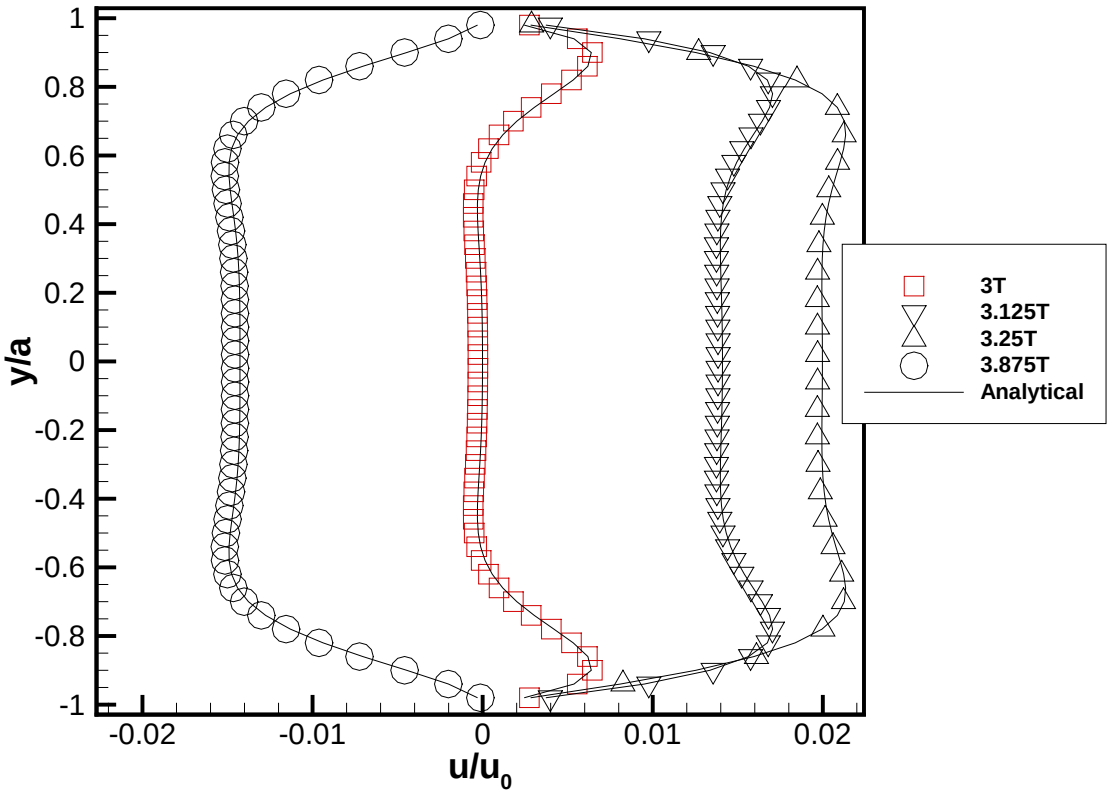


FIGURE 3.32: LBFS $Re = 250, N_y = 50, Wo = 10$, u-velocity profile at $x = L/2$ for Womersley flow at varying times.

results in the most complex velocity profiles. As shown in Figures 3.32 to 3.34 there is excellent agreement between the numerical results generated by the LBFS and the analytical solution. This benchmark problem shows that the LBFS can model unsteady pressure driven flows to a high level of accuracy.

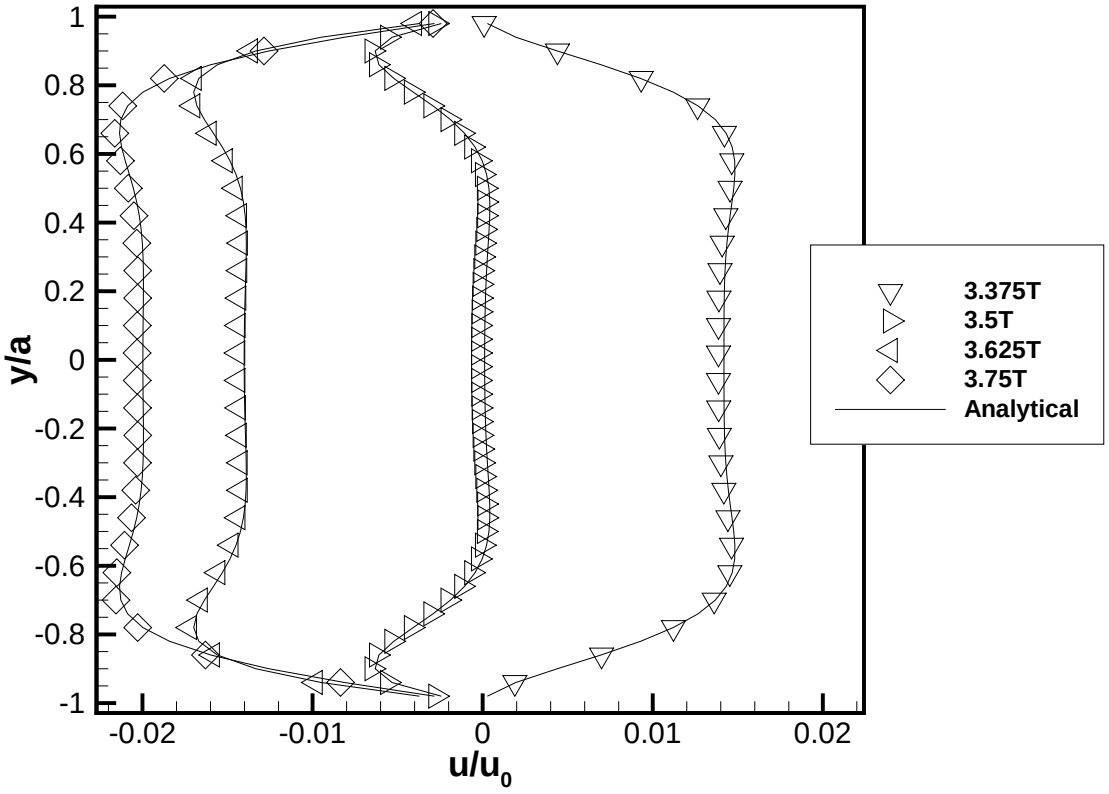


FIGURE 3.33: LBFS $Re = 250, N_y = 50, Wo = 10$, u-velocity profile at $x = L/2$ for Womersley flow at varying times.

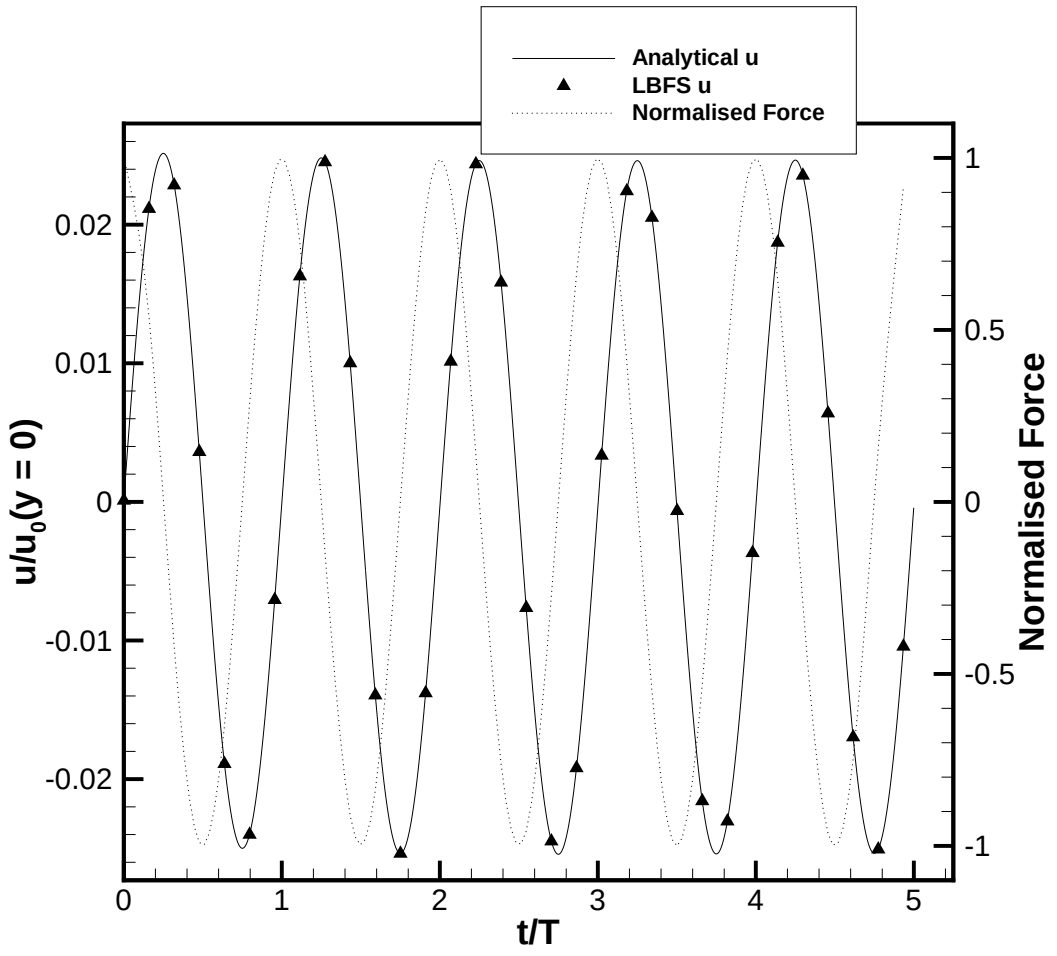


FIGURE 3.34: LBFS $Re = 250, N_y = 50, Wo = 10$, u-velocity at $y = 0$ for Womersley flow at varying times.

3.7 Non-Uniform Grid Lid-Driven Cavity

3.7.1 Introduction

Shear-driven flow in a cavity is a standard flow problem for verifying incompressible viscous flow codes. It has historically been used as the benchmark study to compare the accuracy and the performance of CFD codes [183]. Previously this benchmark problem was run with a uniform grid. In this chapter the numerical experiment is repeated and a non-uniform structured grid has been used. The results show the performance of the LBFS when using non-uniform grids and are subsequently compared to the uniform grid results.

This benchmark problem was also be used to investigate methods of calculating initial values of the macroscopic variables on the localised lattices at the cell interface. Two approaches to this include direct interpolation or calculation from the gradients of the adjacent cells. Direct interpolation for each node was shown to be prohibitive in its computational expense as run times were larger by one order of magnitude. Hence results for two gradient calculation methods, Gauss-Gradient and Least-Squares, are shown in this section.

3.7.2 Problem Set Up

The lid-driven cavity is a steady state problem with a moving lid developing the flow. The LBFS will iterate the solution in time until a steady state is reached. This can then be compared to existing numerical results produced by Ghia et al. [4]. The initial set up of the lid-driven cavity problem is illustrated in Figure 3.35. The top boundary moves with a velocity u_{lid}. This is implemented using a Dirichlet boundary condition specifying u_{lid}. A no-slip boundary condition was implemented at the remaining three boundaries. This was implemented using a Dirichlet boundary condition specifying zero velocity at these boundaries. For pressure, a Neumann boundary condition specifying zero change in pressure across the boundary was implemented at all boundaries.

The fluid in the cavity was chosen as water due to its similar properties to plasma. The test case was run for a range of Reynolds numbers, *i.e.* Re = 100, 400, 1000, 3200, for the physical parameters provided in Table 3.15.

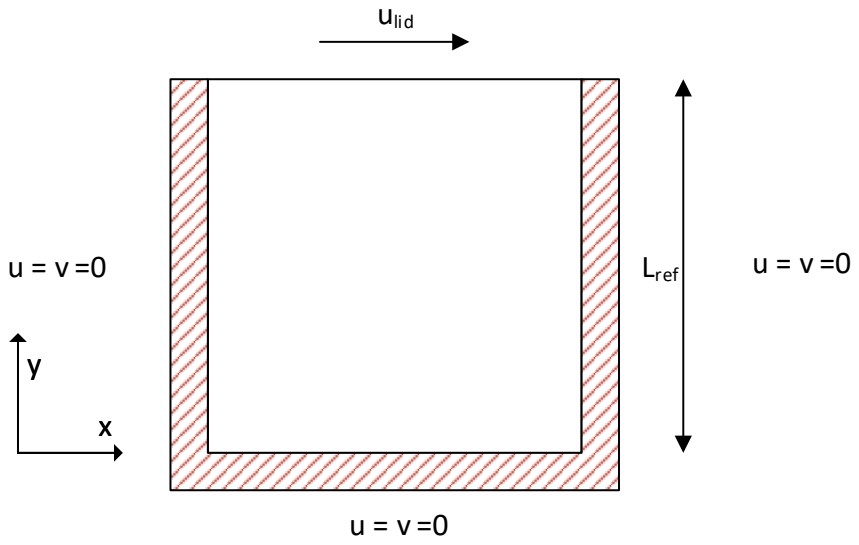


FIGURE 3.35: Lid-driven cavity flow - set up.

Variable Name	Value	Units
L_{ref}	0.1	m
ν	$1.0034 * 10^{-6}$	$\mathrm{m\,s^{-1}}$
ρ	998.2	$\mathrm{kg\,m^{-3}}$

TABLE 3.15: Overall test case parameters for lid-driven cavity flow.

Each test case is run using a non-uniform grid. The non-uniform grid used in each test case can be generated by using the Chebyshev node formulation:

$$x_i = \frac{1}{2}\left[1 - cos\left(\frac{i-1}{N_x - 1}\pi\right)\right] for\ i\ = 1, 2, ..., N_x \tag{3.28}$$

$$y_j = \frac{1}{2}\left[1 - cos\left(\frac{j-1}{N_y - 1}\pi\right)\right] for\ j\ = 1, 2, ..., N_y \tag{3.29}$$

where N_x and N_y are respectively the total number of mesh points in the x and y direction respectively. An example of a 61 x 61 grid is shown in Figure 3.36.

Local timestepping was used to accelerate the convergence of the solution to steady-state. The approach adopted was the same as described in Section 2.5.2 and involves using the largest time step at each cell that is allowed by stability criteria. Each test case was run for both the Gauss-Gradient and Least-Squares approach to gradient calculations which are described in detail in Appendix A.4.

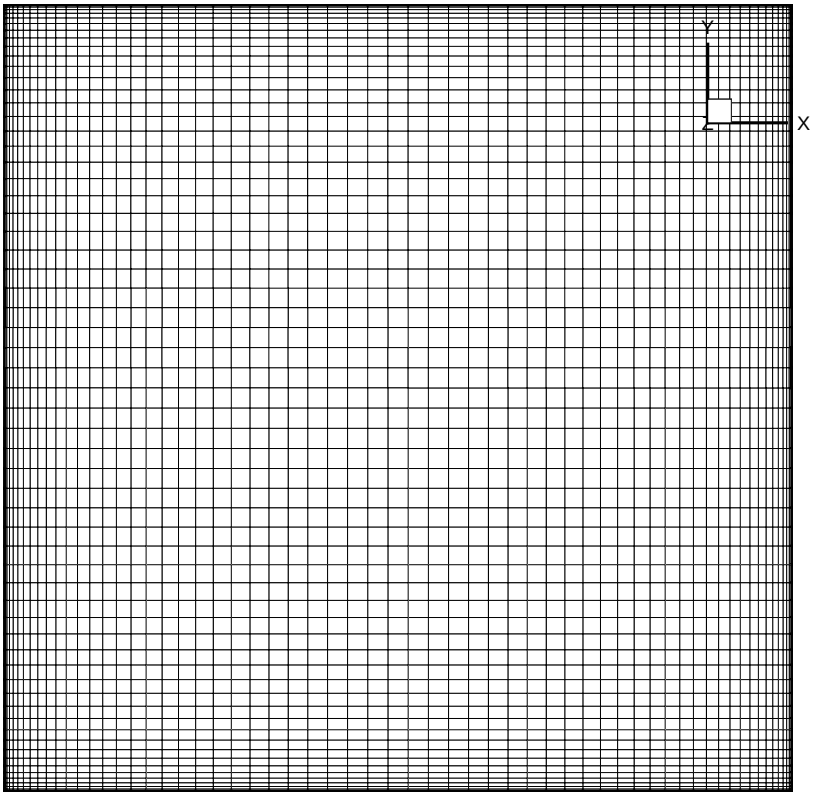


FIGURE 3.36: 61 x 61 non-uniform grid.

The value of the lid velocity changes with Reynolds number and a test case was run for each Reynolds number. The parameters for the test cases are summarised in Table 3.16.

Test Case No.	Re	$u_{lid}(m/s)$	No. of Cells	$N_{x,y}$	$\frac{\Delta x}{\delta x}$	$\Delta t(s)$	Gradient Calc
1	100	0.001	3721	61	2	local	Least-Squares
2	400	0.004	3721	61	2	local	Least-Squares
3	1000	0.010	6561	81	2	local	Least-Squares
4	3200	0.032	10201	101	2	local	Least-Squares
5	100	0.001	3721	61	2	local	Gauss
6	400	0.004	3721	61	2	local	Gauss
7	1000	0.010	6561	81	2	local	Gauss
8	3200	0.032	10201	101	2	local	Gauss

TABLE 3.16: Individual test case parameters varying with Reynolds number for lid-driven cavity flow.

3.7.3 Convergence Criteria

For all test cases, the RMS of the velocity change between successive iterations was used to monitor convergence to steady state. The convergence criteria was defined as follows:

$$R\left(\mathbf{Q}_{Iteration,N}\right) < 1 * 10^{-6} \tag{3.30}$$

where:

$$R\left(\mathbf{Q}_{Iteration,N}\right) = \sum_{1}^{No\ of\ Cells} \sqrt{\frac{\left(\left(\sqrt{u^2+v^2}\right)^{N+1} - \left(\sqrt{u^2+v^2}\right)^{N}\right)^2}{\left(\left(\sqrt{u^2+v^2}\right)^{N+1}\right)^2}} \tag{3.31}$$

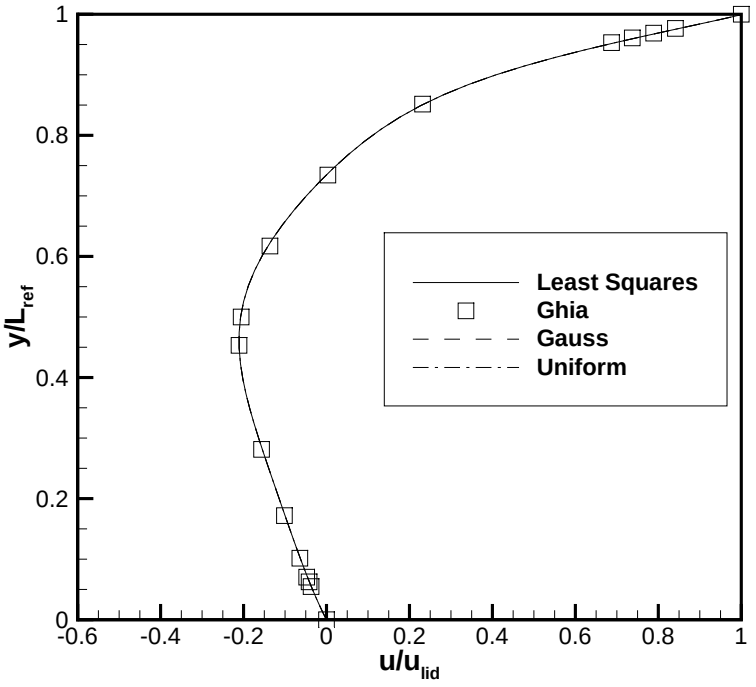
and N is the number of iterations completed in the solution. This measure of residual is used by Wang et al. [136].

3.7.4 Results

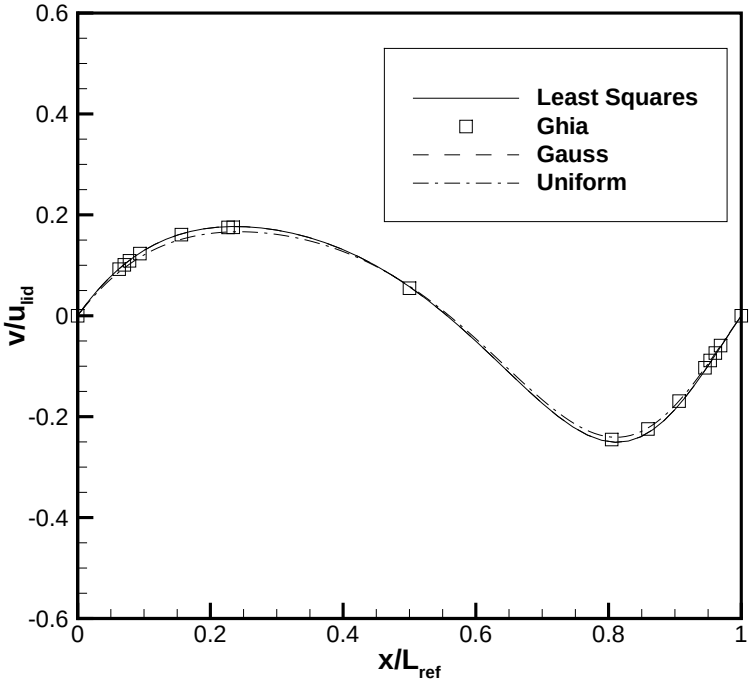
The results generated by the LBFS were compared to the numerical results produced by Ghia et al. [4]. The following aspects of the simulation are compared:

- Centre line velocity plots.
- Streamline plots.
- Primary vortex location.
- Run time comparison.

The velocity profiles of the steady state solution were calculated across the centrelines of the domain in the x and y direction. The results for Test Cases 1-8 are shown in Figures 3.37 to 3.40.

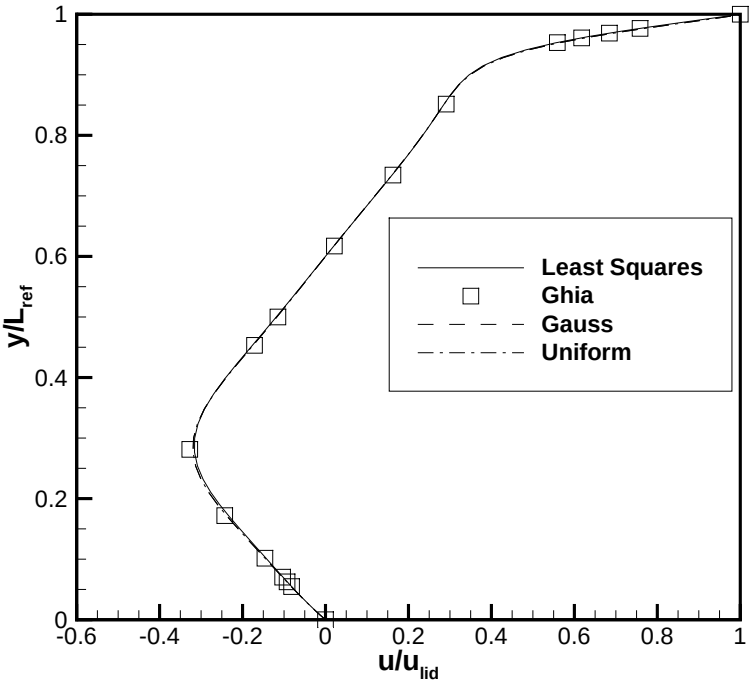


(a)

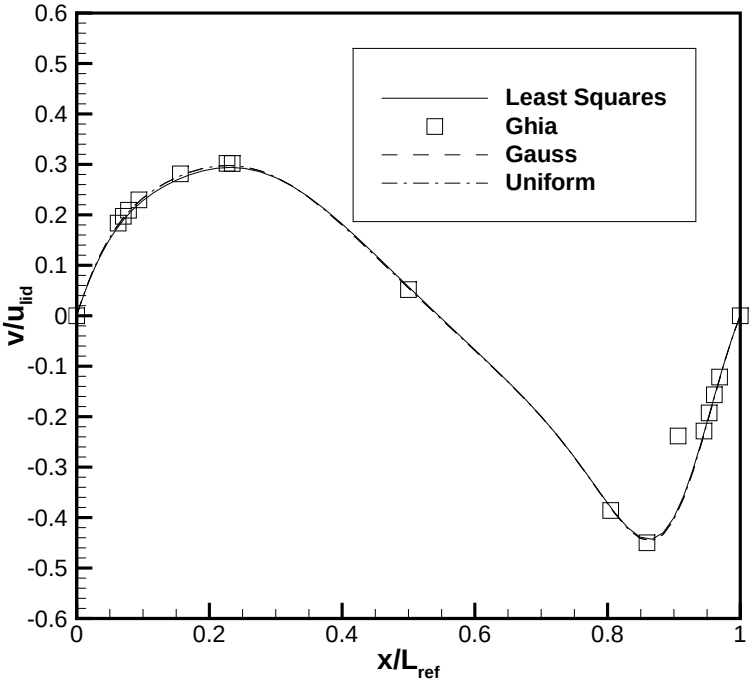


(b)

FIGURE 3.37: Non-uniform grid lid-driven cavity flow centreline velocity profiles, Re =100.

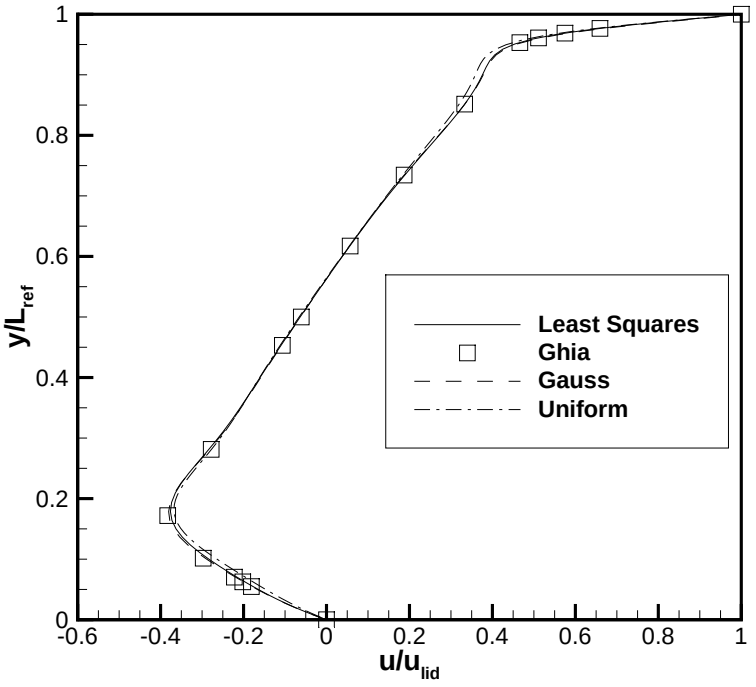


(a)

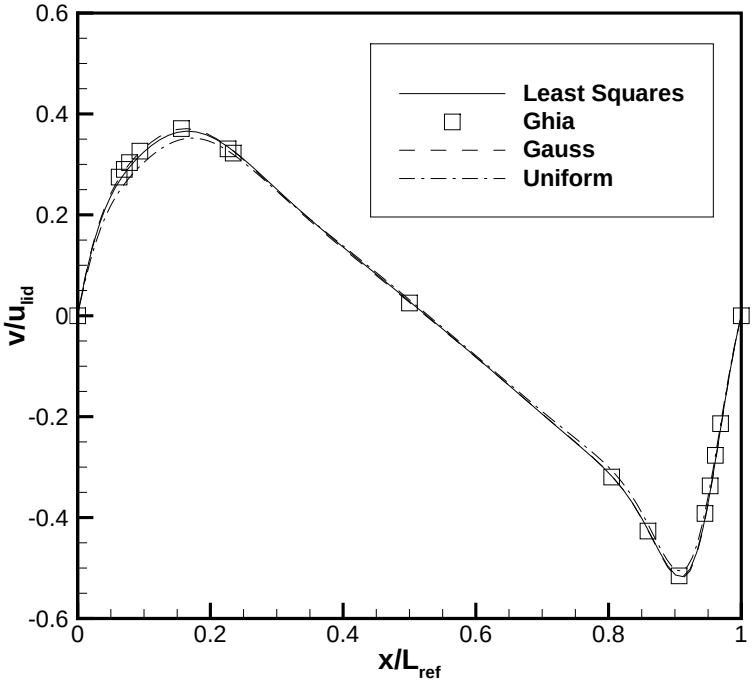


(b)

FIGURE 3.38: Non-uniform grid lid-driven cavity flow centreline velocity profiles, Re =400.

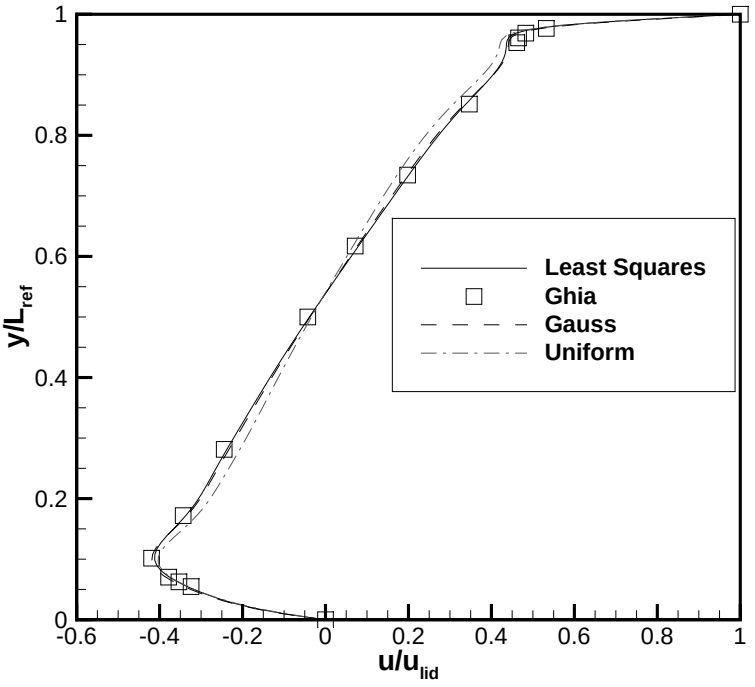


(a)

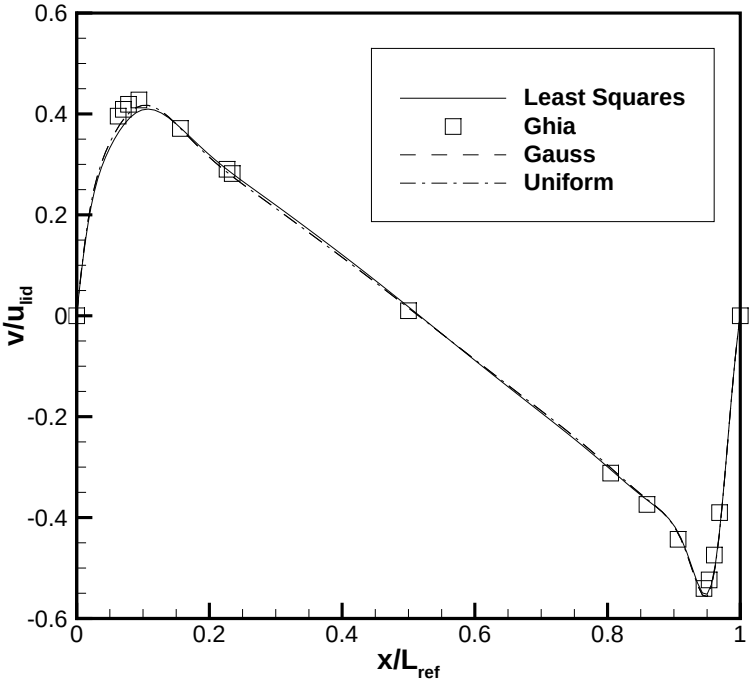


(b)

FIGURE 3.39: Non-uniform grid lid-driven cavity flow centreline velocity profiles, Re =1000.



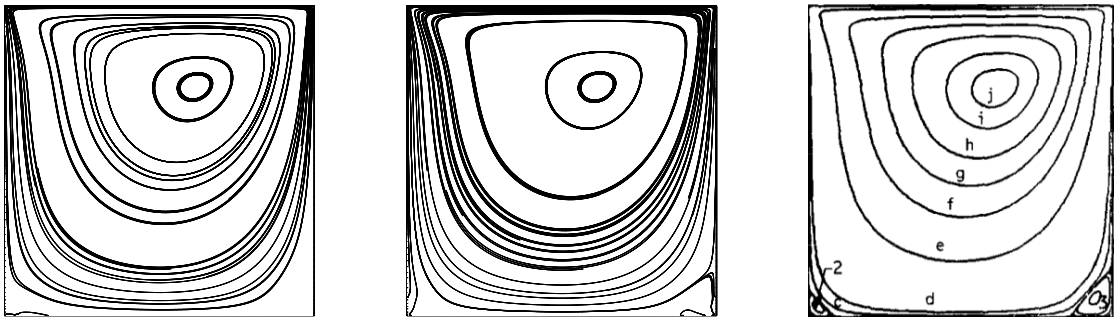
(a)



(b)

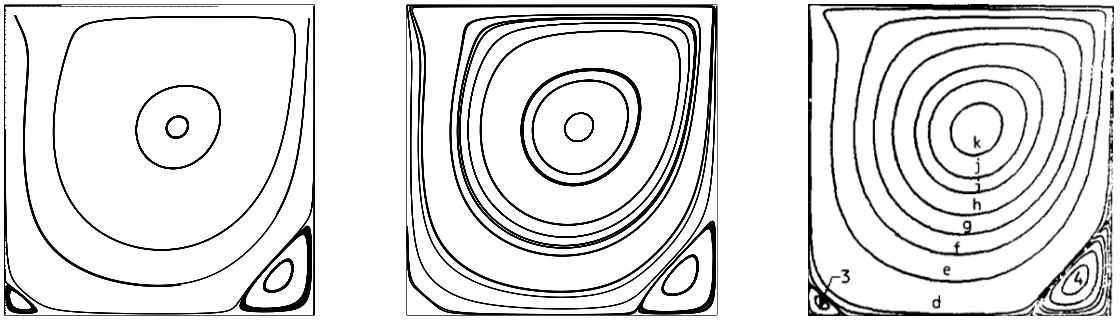
FIGURE 3.40: Non-uniform grid lid-driven cavity flow centreline velocity profiles, Re =3200.

The streamline plots for each of the test cases are plotted along with the results of Ghia et al. [4] in Figures 3.41 to 3.44.



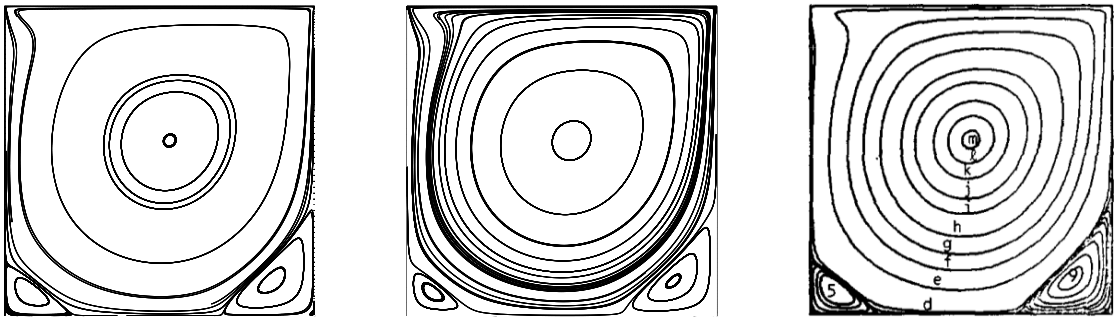
(a) Least-Squares (b) Gauss (c) Ghia

FIGURE 3.41: Comparison of streamline plots for lid-driven cavity flow between LBFS (non-uniform grid) and Ghia for Re = 100.



(a) Least-Squares (b) Gauss (c) Ghia

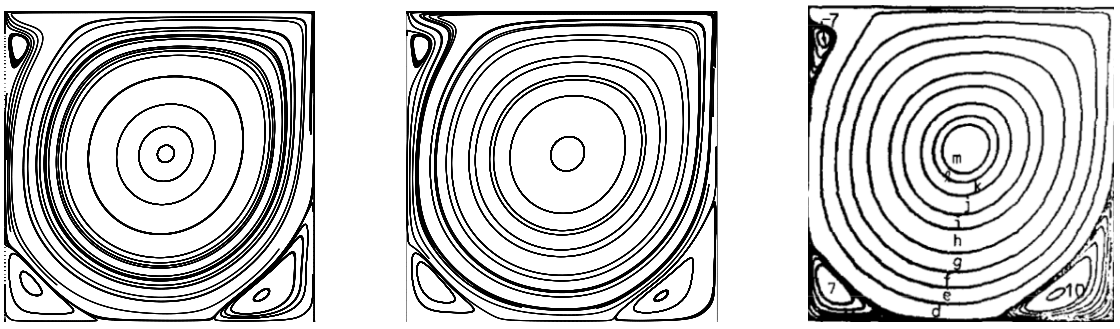
FIGURE 3.42: Comparison of streamline plots for lid-driven cavity flow between LBFS (non-uniform grid) and Ghia for Re = 400.



(a) Least-Squares (b) Gauss (c) Ghia

FIGURE 3.43: Comparison of streamline plots for lid-driven cavity flow between LBFS (non-uniform grid) and Ghia for Re = 1000.

The location of the centre of the primary vortex was taken as the point with the minimum value of the streamfunction. It is then compared to results provided by Ghia et al. [4] and LBFS with uniform grid in Table 3.17.



(a) Least-Squares (b) Gauss (c) Ghia

FIGURE 3.44: Comparison of streamline plots for lid-driven cavity flow between LBFS (non-uniform grid) and Ghia for Re = 3200.

Next the simulation run times are compared for the different methods. Run times are compared for both the Least-Squares and Gauss-Gradient methods. These are also compared to similar test cases run using a uniform grid. These are shown in Table 3.18.

3.7.5 Conclusions

The results demonstrated above show that the LBFS can calculate the steady state solution of the lid-driven cavity problem to a very high level of accuracy using non-uniform grids. Previously uniform grids were used, using finer meshes and the results using a non-uniform grid are superior in accuracy and run time. These results are also comparable to the accuracy achieved by Shu et al. [137].

A comparison between the Gauss-Gradient and Least-Squares approaches for calculating gradients was also done. It was found that both approaches resulted in very accurate results. The Gauss-Gradient approach seems to be marginally more accurate at high Reynolds numbers. It also seems to be approximately ten percent faster for equivalent simulations.

However this is as expected as the Gauss-Gradient approach is optimal on grids with very low skewness. With the structured non-uniform mesh used, there was no requirement to use a modification term to allow for errors due to skewness.

This flow problem has demonstrated that the LBFS can handle the use of non-uniform grids to a high degree of accuracy. This enables the use of refined meshes which allow increased accuracy or reduced runtimes over an equivalent uniform mesh.

Re	Vortex Centre (x,y)				
	No. of Cells	Ghia et al.[4]	Gauss	Least-Squares	Uniform Grid
100	3721	(0.6172,0.7344)	(0.6147,0.7324)	(0.6171,0.7347)	-
400	3721	(0.5547,0.6055)	(0.5572,0.6039)	(0.5572,0.6011)	-
1000	6561	(0.5313,0.5625)	(0.5320,0.5635)	(0.5343,0.5635)	-
3200	10201	(0.5165,0.5469)	(0.5169,0.5378)	(0.5203,0.5378)	-
100	16641	(0.6172,0.7344)	-	-	(0.6208,0.7460)
400	16641	(0.5547,0.6055)	-	-	(0.5641,0.6153)
1000	16641	(0.5313,0.5625)	-	-	(0.5357,0.5721)
3200	16641	(0.5165,0.5469)	-	-	(0.5185,0.5450)

TABLE 3.17: Co-ordinates of primary vortex centres in lid-driven cavity flow for individual test cases.

Re	Simulation Run Times (s)			
	No. of Cells	Gauss	Least-Squares	Uniform
100	3721	153.24	148.33	-
400	3721	177.09	192.25	-
1000	6561	500.57	542.68	-
3200	10201	1219.71	1366.36	-
100	16641	-	-	1177.27
400	16641	-	-	2704.46
1000	16641	-	-	3984.87
3200	16641	-	-	8463.08

TABLE 3.18: Run time comparison for lid-driven cavity flow between Gauss-Gradient, Least-Squares and uniform grid approaches.

3.8 Summary

In this chapter the solutions of a large variety of complex flow problems have been predicted by the LBFS. It has been fully verified in 2D that the LBFS can handle the following features of fluid flow problems:

1. Can correctly calculate the viscous and inviscid fluxes for a variety of Reynolds numbers.
2. Can predict accurate results for both steady and unsteady flow problems.
3. Can accurately predict the outcomes of pressure driven flow.
4. Has demonstrated that the LBFS has between first and third order accuracy for 2D flow problems.
5. Can be used as a solver with non-uniform meshes with a high degree of accuracy, enabling either an increase in accuracy or reduced runtimes when compared to using an equivalent uniform mesh.

The results in this chapter show that the LBFS is a viable N-S solver and therefore the effort to extend the LBFS to 3D and to unstructured grids can be justified. The verification of these further extensions in functionality will be discussed in the next chapter.

Chapter 4

Three-Dimensional Preconditioned Lattice Boltzmann Flux Solver on Unstructured Grids

4.1 Introduction

As discussed in Section 1.8, the LBFS performs poorly at incompressible steady-state problems. In this chapter the γ-preconditioning approach is introduced and applied to two benchmark flow problems. The results and outcomes of this implementation are then discussed. In Chapter 3 the LBFS's ability to predict 2D flow problems on non-uniform grids was verified. This chapter will also show the results of extending the LBFS to 3D and extending the meshes used from non-uniform to fully unstructured hexahedral meshes.

4.2 Governing Equations

The preconditioned lattice Boltzmann flux solver (PLBFS) takes the same form as the LBFS with the preconditioned flux tensor $\mathbf{F}_p$ given by:

$$\begin{aligned}\mathbf{F}_p = &\sum_{\alpha=0}^{N} f_\alpha^{eq} \begin{bmatrix} e_{\alpha,x} & e_{\alpha,y} & e_{\alpha,z} \\ e_{\alpha,x}e_{\alpha,x} & e_{\alpha,y}e_{\alpha,x} & e_{\alpha,z}e_{\alpha,x} \\ e_{\alpha,x}e_{\alpha,y} & e_{\alpha,y}e_{\alpha,y} & e_{\alpha,z}e_{\alpha,y} \\ e_{\alpha,x}e_{\alpha,z} & e_{\alpha,y}e_{\alpha,z} & e_{\alpha,z}e_{\alpha,z} \end{bmatrix} \\ &+ \sum_{\alpha=0}^{N} \left[1 - \frac{1}{2\tau_p}\right] f_\alpha^{neq} \begin{bmatrix} 0 & 0 & 0 \\ e_{\alpha,x}e_{\alpha,x} & e_{\alpha,y}e_{\alpha,x} & e_{\alpha,z}e_{\alpha,x} \\ e_{\alpha,x}e_{\alpha,y} & e_{\alpha,y}e_{\alpha,y} & e_{\alpha,z}e_{\alpha,y} \\ e_{\alpha,x}e_{\alpha,z} & e_{\alpha,y}e_{\alpha,z} & e_{\alpha,z}e_{\alpha,z} \end{bmatrix}\end{aligned} \tag{4.1}$$

where f_α^{eq} is defined as:

$$f_\alpha^{eq}(\mathbf{r},t) = \rho w_\alpha \left[1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{V}}{c_s^2} + \frac{(\mathbf{e}_\alpha \cdot \mathbf{V})^2 - (c_s|\mathbf{V}|)^2}{2\gamma c_s^4}\right] + O(\mathbf{V}^3) \tag{4.2}$$

and f_α^{neq} is defined as:

$$f_\alpha^{neq}(\mathbf{r},t) = -\tau_p \left[f_\alpha^{eq}(\mathbf{r},t) - f_\alpha^{eq}(\mathbf{r} - \mathbf{e}_\alpha \delta t, t - \delta t)\right] \tag{4.3}$$

where γ is the preconditioning parameter and τ_p is the preconditioned relaxation factor. The preconditioning parameter γ relates τ_p to the standard relaxation factor τ_s in the following way:

$$\gamma = \frac{\tau_s - 0.5}{\tau_p - 0.5} \tag{4.4}$$

Letting the preconditioning matrix $\mathbf{P} = \ diag\,(1, \gamma^{-1}, \gamma^{-1}, \gamma^{-1})$, then through the Chapman-Enskog expansion, the flux tensor in Equation (2.1) for the preconditioned NS equations can be shown to be:

$$\mathbf{F}_p = \mathbf{P} \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + \overline{p} - \mu\left(2\frac{\partial u}{\partial x}\right) & \rho uv - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \rho uw - \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \rho vu - \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \rho v^2 + \overline{p} - \mu\left(2\frac{\partial v}{\partial y}\right) & \rho vw - \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \rho wu - \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) & \rho wv - \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) & \rho w^2 + \overline{p} - \mu\left(2\frac{\partial w}{\partial z}\right) \end{bmatrix} \tag{4.5}$$

where:

$$\overline{p} = \gamma c_s^2 \rho \tag{4.6}$$

and:

$$\mu = \gamma \rho c_s^2 \left(\tau_p - 0.5\right) \delta t \tag{4.7}$$

For steady flows utilising Equation (4.5) in Equation (2.1) is equivalent to utilising Equation (2.2) in Equation (2.1) but with a different equation of state. To demonstrate the effect of the preconditioning parameter on the rate of convergence, the partial differential equation form of the preconditioned NS equations is employed:

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{PA}\frac{\partial \mathbf{Q}}{\partial x} + \mathbf{PB}\frac{\partial \mathbf{Q}}{\partial y} + \mathbf{PC}\frac{\partial \mathbf{Q}}{\partial z} = 0 \tag{4.8}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are the Jacobians of the flux vectors given by:

$$\mathbf{A} = \frac{\partial \mathbf{F}_{p,x}}{\partial \mathbf{Q}} \ , \ \mathbf{B} = \frac{\partial \mathbf{F}_{p,y}}{\partial \mathbf{Q}} \ , \ \mathbf{C} = \frac{\partial \mathbf{F}_{p,z}}{\partial \mathbf{Q}} \tag{4.9}$$

and $\mathbf{F}_{p,x}$, $\mathbf{F}_{p,y}$, $\mathbf{F}_{p,z}$ are the components of the flux tensor in the x, y and z directions respectively. Considering only the inviscid terms, these are given by:

$$\mathbf{F}_{p,x} = \begin{bmatrix} \rho u \\ \rho u^2 + \overline{p} \\ \rho v u \\ \rho w u \end{bmatrix} \ , \ \mathbf{F}_{p,y} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + \overline{p} \\ \rho w v \end{bmatrix} \ , \ \mathbf{F}_{p,z} = \begin{bmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + \overline{p} \end{bmatrix} \tag{4.10}$$

The preconditioned convection matrix $\mathbf{PA}$ is now written as:

$$\mathbf{PA} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\gamma} & 0 & 0 \\ 0 & 0 & \frac{1}{\gamma} & 0 \\ 0 & 0 & 0 & \frac{1}{\gamma} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ \gamma c_s^2 - u^2 & 2u & 0 & 0 \\ -vu & v & u & 0 \\ -wu & w & 0 & u \end{bmatrix} \tag{4.11}$$

To calculate the eigenvalues of $\mathbf{PA}$, the determinant of $(\mathbf{PA} - \lambda \mathbf{I})$ is first calculated:

$$det\left(\mathbf{PA} - \lambda \mathbf{I}\right) = \frac{\left(u - \lambda\right)^2}{\gamma^3} \left[\lambda^2 - 2u\lambda + u^2 - \gamma c_s^2\right] \tag{4.12}$$

The eigenvalues λ are found by letting Equation (4.12) equal 0 giving:

$$\lambda\left(\mathbf{PA}\right) = \frac{1}{\gamma}\left(u, u, u \pm \gamma c_s\right) \tag{4.13}$$

The aim of preconditioning is to reduce the stiffness of the system by scaling the eigenvalues of **PA** appropriately. The reduction in stiffness can be measured by the condition number (CN) given as:

$$CN(\mathbf{PA}) = \frac{|max\left[\lambda\left(\mathbf{PA}\right)\right]|}{|min\left[\lambda\left(\mathbf{PA}\right)\right]|} = 1 + \frac{\gamma c_s}{u} = 1 + \frac{\gamma}{Ma} \tag{4.14}$$

where Ma is the dimensionless Mach number. A CN close to 1 indicates a well balanced system with no stiffness. From Equation (4.14) it can be seen that to achieve a CN close to 1, it is required that $Ma \rightarrow \infty$ or $\gamma \rightarrow 0$. As Ma is restricted to values < 0.4 to satisfy incompressiblity requirements, reducing γ can be used to reduce the stiffness of the system. The above only considers one dimension, in reality a flow will have three dimensional flow. To consider the impact of preconditioning on three dimensional flow; the CN of **PB** and **PC** can be calculated following the same approach.

4.3 Impact of Preconditioning on Unstructured Grids

4.3.1 Overview

One advantage of using an unstructured grid with the LBFS is that it allows local refinement of the mesh in areas of rapidly changing flow. In practice, this involves having a very fine mesh close to surfaces in the flow and a much coarser mesh further away. This is in contrast to the traditional LBM where a uniform mesh density is used throughout the computational domain. When Guo [158] applied preconditioning to the LBM, there was no need to consider the impact of preconditioning on larger convection dominated cells or relatively finer cells in the boundary layer that are viscous dominated. As a result the purpose of that work was to reduce the stiffness of the inviscid Jacobians. In this section it will be shown that the PLBFS can improve the accuracy of flux calculations in larger cells compared to the LBFS but can also have an adverse effect on local timestepping in viscous dominated cells. An optimal choice of γ will consider both these factors while trying to reduce the CN of the system, and a strategy for making this choice is proposed at the end of this section.

4.3.2 Influence of Preconditioning on Lattice Streaming Distance

When choosing a lattice streaming distance δx, the preferred strategy is to maximise the size of the lattice while keeping all lattice nodes within the cells adjoining to the common cell interface. This has the effect of reducing the error in interpolating the macroscopic variables to the lattice nodes using Equation (2.13) and Equation (2.14). This is due to the fact that as δx increases, the distance between the lattice nodes and the cell centre, from which the macroscopic variables are interpolated, is reduced. This increases the accuracy of the interpolation as the gradient is calculated at the cell centre and the lattice nodes are closer to the cell centre.

However this leads to issues in relatively large cells that are prevalent in unstructured grids. From Equation (4.7), as δx increases τ_p decreases. This has an adverse affect on stability as the LBM becomes unstable as τ_p tends to its minimum value 0.5 especially in the range $0.50 < \tau_p < 0.55$ [187]. Preconditioning is most advantageous in scenarios when the stability limit of τ_p limits the size of δx. Decreasing γ allows an increase in δx for a constant τ_p. The benefits are illustrated by Figure 4.1, where decreasing γ allows a much larger lattice reconstruction without introducing instability by lowering the value of τ_p.

As result of this property, preconditioning allows the use of coarser unstructured grids than the unpreconditioned LBFS with increased accuracy. This allows for either a decrease in runtimes for similar levels of accuracy or an increase in accuracy for the same runtime.

4.3.3 Influence of Preconditioning on Viscous Dominated Cells

Using preconditioning can have an adverse effect on timestepping in viscous dominated cells, *i.e.* where the viscous spectral radii is larger than the convective spectral radii, or can cause convection dominated cells to become viscous dominated. To illustrate the impact of γ on the local time step of these relatively fine cells employed in viscous dominated regions, the spectral radii of a cell is calculated using Equations (2.55) to (2.58) and it's assumed that $\mathbf{V} \to 0$ as the cell is

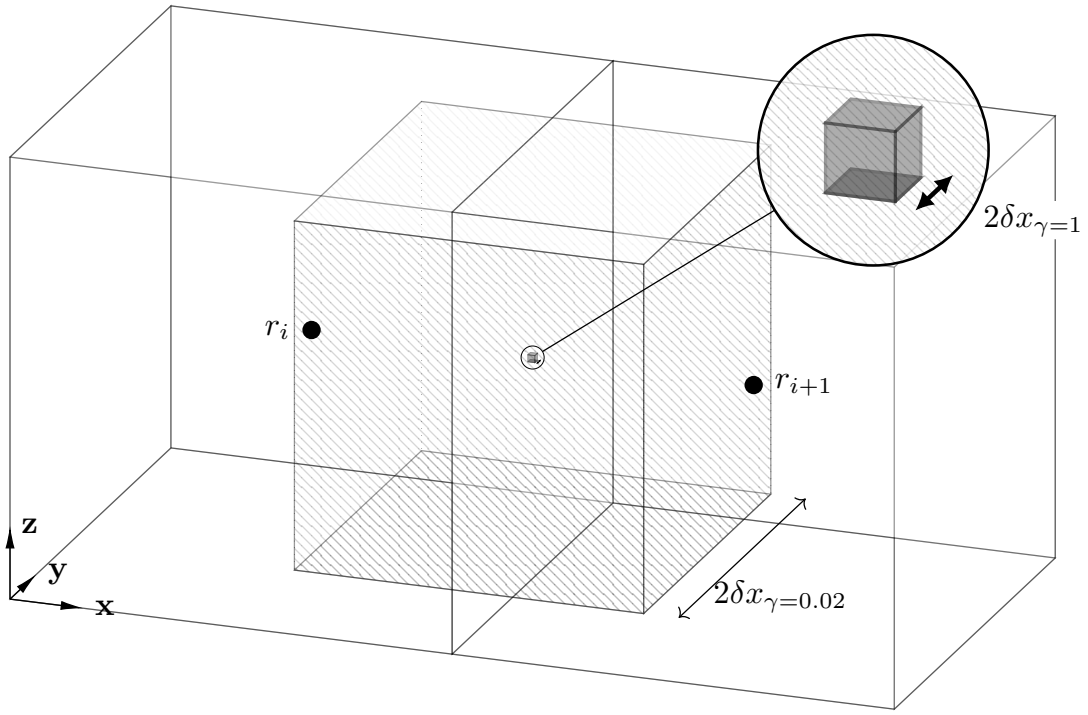


FIGURE 4.1: Local reconstruction of the lattice Boltzmann lattice for $\gamma = 1$ and $\gamma = 0.02$ where $Re = 1000$, $Ma = 0.17$ and $\mu = 0.0001$.

assumed to neighbour a surface in the flow. For a uniform cell with edge length l, surface area S and volume V the convective and viscous spectral radii become:

$$\begin{aligned} \Lambda_c &= \frac{1}{\gamma} 3\left(0 + \gamma c_s\right) S = 3 c_s S \\ \Lambda_v &= \frac{1}{\gamma V} \frac{2\mu}{\rho} \cdot 3S^2 = \frac{6\mu l}{\gamma \rho} \end{aligned} \tag{4.15}$$

The ratio of the viscous spectral radius to the inviscid spectral radius, including the constant in Equation (2.55), is given as:

$$\Gamma_{vc} = \frac{C\Lambda_v}{\Lambda_c} = 4\frac{2\mu}{\gamma c_s l \rho} \tag{4.16}$$

Re-writing in non-dimensional form and defining l relative to the reference length L gives:

$$\Gamma_{vc} = \frac{8MaN}{\gamma Re} = \Phi N \tag{4.17}$$

where Φ is a constant and $N = L/l$ is the number of uniform cells along the reference length L. Values of $\Gamma_{vc} > 1$ indicate that the flow in a cell is viscous dominated. If $\Gamma_{vc} >> 1$ this means that the time step in Equation (2.55) is calculated based off the viscous spectral radius and is not optimised for the inviscid

contributions. Figure 4.2 illustrates how Φ and cell size influence if a cell is convection or viscous dominated. Cells becomes viscous dominated as the edge length of the cell becomes smaller and the viscous threshold of edge length, where the cell changes from convection dominated to viscous dominated, is dependent on Φ. Smaller values of Φ allow finer cells to remain convection dominated. To reduce the value of Φ, the following choice of non-dimensional parameters is desired:

- Reduced values of Ma.
- Larger values of γ.
- Larger values of Re.

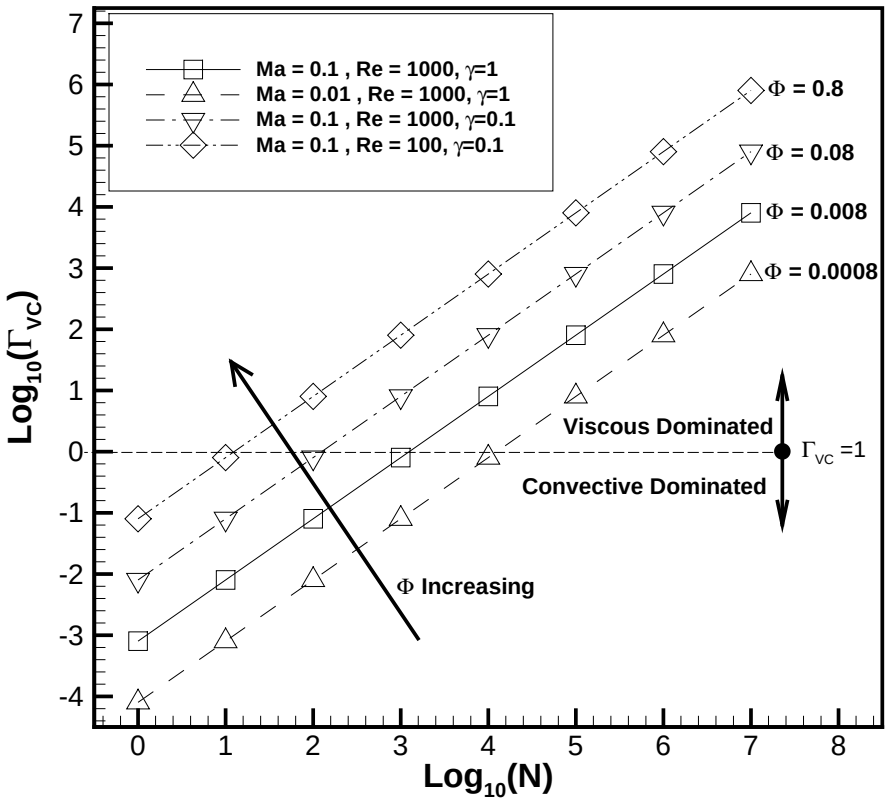


FIGURE 4.2: Variation of Γ_{vc} with N for different values of Φ.

4.3.4 Preconditioning Parameter Selection Strategy

The purpose of preconditioning the N-S equations is to reduce the stiffness of the inviscid Jacobians to accelerate convergence of steady flow calculations. However, as shown in the previous sections, the use of preconditioning can have advantages and disadvantages when it comes to the use of unstructured grids. Preconditioning allows the use of larger cells in convection dominated flow which leads to reduced runtimes. However if the mesh contains a large number of viscous dominated cells, in particular cells in the boundary layer of a surface, this may have an adverse impact on the convergence rate. A good choice of preconditioning parameter would have $1+\frac{\gamma}{Ma} \rightarrow 1$ whilst keeping $\Gamma_{vc} \leq 1$ for the vast majority of cells. From Equation (4.17) and Figure 4.2 it can be seen that smaller values of Ma and larger values of Re and γ allow smaller cells to be used while keeping $\Gamma_{vc} \leq 1$. Setting $\Gamma_{vc} = 1$ in Equation (4.17) gives the following equation for setting γ:

$$\gamma > \frac{8MaN}{Re} \tag{4.18}$$

Note that Equation (4.18) will only hold for every cell in uniform grids only. In grids where there is a large variation in cell size, there will be cells that will have $\Gamma_{vc} > 1$. Therefore it is recommended to investigate a histogram of cell size of the computational domain and optimise γ for the most common cell size.

The impact of the choice of preconditioning parameter on stability should also be considered. Izquierdo suggested the optimal choice of $\gamma = 2Ma$ [161] as this value maximises speed up while offering robust stability. However it was provided with the caveat that the optimal value increases for lower Reynolds numbers. A strategy for choosing a value of γ is given in Table 4.1. It is preferable to decrease Ma rather than have $\Gamma_{vc} > 1$ as lower values of Ma reduces the compressibility error at no cost in runtime when local timestepping is used. It should be noted that accurately calculating the real eigenvalues in a cell is quite difficult and that Equation (4.17) is based on an estimate and that $\Gamma_{vc} = 1$ may not be the exact threshold for cells becoming viscous dominated.

Preconditioning Parameter Selection	
1.	First attempt using the optimal precondition parameter provide by Izquierdo [161] where $\gamma = 2Ma$.
2.	If no convergence acceleration is experienced, then for a given Re and mesh, investigate the histogram of cell sizes in the computational domain and identify the most common cell size.
3.	Using this cell size calculate Γ_{vc} using Equation (4.17) for uniform grids or Equations (2.56) to (2.58) for unstructured grids.
4.	If $\Gamma_{vc} > 1$, then decrease Ma, increase γ or increase the size of the smallest cell in the mesh.
5.	Repeat Step 3 until an acceleration in runtime is achieved or $\gamma = 1$

TABLE 4.1: Strategy for choosing values for preconditioning on unstructured grids.

4.4 Numerical Results and Discussion

4.4.1 Overview

To demonstrate the capability of the PLBFS, two flow problems are considered: 3D lid-driven cavity flow and 3D flow over a circular cylinder. For both flow problems unstructured hexahedral meshes were used to demonstrate that the PLBFS can be effectively employed with an unstructured mesh topology. Flow problems were solved using a variety of Re and Ma numbers and the impact of various values of the preconditioning parameter γ were investigated to show the impact of preconditioning on accuracy and convergence rates. All meshes were created with ANSYS ICEM meshing software (ANSYS Inc., Canonsburg, PA, USA).

The RMS of the ρu-momentum residual was used to monitor convergence to steady-state and is defined as follows:

$$R_{RMS}\left(\mathbf{Q}^{Iteration,t}\right) = \sqrt{\frac{\sum_{i=1}^{No\ of\ Cells} \mathbf{R}\left(\mathbf{Q}_{i,t}\right)^2}{No\ of\ Cells}} \tag{4.19}$$

where t is the iteration index and $\mathbf{R}\left(\mathbf{Q}_i\right)$ is the ρu residual calculated for cell i using Equation (2.26). The calculated RMS was normalised relative to the initial

RMS value calculated at the end of the first iteration. *i.e*

$$R_{RMS,Norm} = \frac{R_{RMS}\left(\mathbf{Q}^{Iteration,t}\right)}{R_{RMS}\left(\mathbf{Q}^{Iteration,1}\right)} \tag{4.20}$$

The convergence criterion employed was a five order magnitude reduction in the normalised residual, *i.e.*

$$R_{RMS,Norm} < 1 * 10^{-5} \tag{4.21}$$

4.4.2 3D Lid-Driven Cavity Flow

Shear-driven flow in a square/cube cavity is a standard test case for validating predictions of incompressible viscous flow. It has historically been used as a benchmark flow problem to investigate the accuracy and the performance of CFD codes [183]. Lid-driven cavity flow is steady for Reynolds number less than 10000, with a moving lid developing the flow. The steady-state predictions by the PLBFS are compared to existing numerical results produced by Ku et al. [188] and Ding et al. [189]. The set up of the lid-driven cavity problem is illustrated in Figure 4.3. The top boundary moves with a velocity u_{lid}. This was implemented using a Dirichlet boundary condition specifying u_{lid}. A no-slip boundary condition was implemented at the remaining five boundaries. For density, a Neumann boundary condition, specifying zero change in density across the boundary, was implemented at all boundaries. The initial conditions were set as $\mathbf{V} = 0$ and $\rho = 1$. Multiple test cases were run for this flow problem with different combinations of Re number, Ma number, preconditioning parameter γ and mesh density. The test case parameters are summarised in Table 4.2 and were chosen as per Table 4.1. Test Cases 1-8 were run using the mesh shown in Figure 4.3 consisting of 90480 hexahedral cells. Test Cases 9-13, performed without preconditioning, were run on finer meshes consisting of 226981 and 531411 cells. The origin of the coordinate system is located at the centroid of the cube and the edges on the cube have a reference length $L_{ref} = 10$. Predicted velocity profiles on vertical and horizontal centrelines respectively in the $z = 0$ plane for Test Cases 1-13 are shown in Figures 4.4 to 4.7. The results are compared to the predictions of Ku et al. [188] and Ding et al. [189]. Overall there is good agreement with the literature by the preconditioned test cases and the test cases performed on finer meshes. To show the flow patterns of the 3D lid-driven cavity flow, 2D streamlines are projected onto three centroidal planes at $x = 0$, $y = 0$ and $z = 0$. The flow patterns are

Test Case No.	u_{lid}	Re	Ma	γ	CN	$\Gamma_{vc,max}$	δx_{max}	N_{cells}
1	0.1	100	0.173	1	6.77	0.622	0.60	90480
2	0.1	100	0.173	0.2	2.15	3.11	3.00	90480
3	0.01	100	0.017	1	58.73	0.06	0.06	90480
4	0.01	100	0.017	0.02	2.15	3.11	3.00	90480
5	0.1	1000	0.173	1	6.77	0.062	0.06	90480
6	0.1	1000	0.173	0.2	2.15	0.311	0.30	90480
7	0.01	1000	0.017	1	58.73	0.006	0.006	90480
8	0.01	1000	0.017	0.02	2.15	0.311	0.30	90480
9	0.1	100	0.173	1	6.77	0.854	0.6	226981
10	0.01	100	0.017	1	58.73	0.085	0.06	226981
11	0.1	1000	0.173	1	6.77	0.085	0.06	226981
12	0.01	1000	0.017	1	58.73	0.008	0.006	226981
13	0.01	1000	0.017	1	58.73	0.011	0.006	531411

TABLE 4.2: Individual test case parameters for 3D lid-driven cavity flow.

shown for Test Cases 4 and 8 where $Re = 100$ and $Re = 1000$ respectively. These results are shown in Figure 4.8 and show the development of stronger secondary vortices and a stronger 3D impact as the Reynolds number increases. This is in agreement with the predictions of previous studies by Ku et al. [188], Ding et al. [189] and Wang et al. [140]. Convergence histories are plotted in Figure 4.9 and Figure 4.10.

Using an unstructured hexahedral mesh produces excellent results that agree with predictions by other researchers in the literature. The previous study using a LBFS by Wang et al. [140] used a $81X81X81$ (531441 cells) non-uniform structured mesh and its predictions had excellent agreement with the studies of Ku et al. [188] and Ding et al. [189]. Using the PLBFS on a fully unstructured grid allows similar levels of accuracy to be attained while using a mesh with only 90480 cells. The results also confirm the theoretical analysis in Section 4.3.3. This analysis suggests that preconditioning should enable the use of large δx on lattices in relatively larger cells. This should reduce the interpolation error when applying Equations (2.13) and (2.14). Inspecting Table 4.2 shows that preconditioning allows the use of a larger δx in the bigger cells in the mesh. Inspecting Figures 4.4 to 4.7 show that the preconditioned case with the larger δx is more accurate than the unpreconditioned cases with smaller δx. Figures 4.4 to 4.7 also show that increasing the mesh density results in more accurate comparisons with the benchmark solutions. There is also significant convergence acceleration with the use of preconditioning at lower Ma

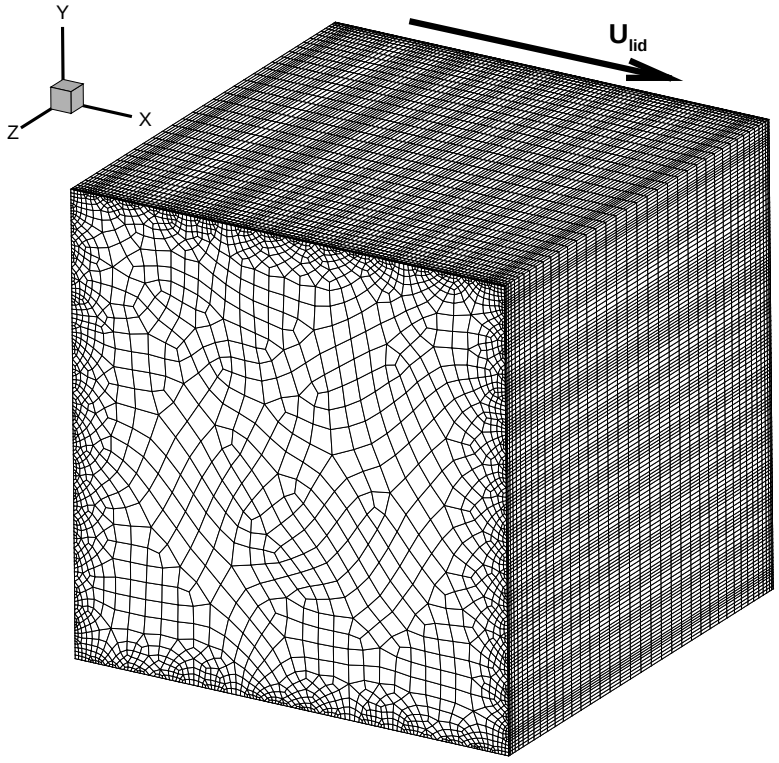
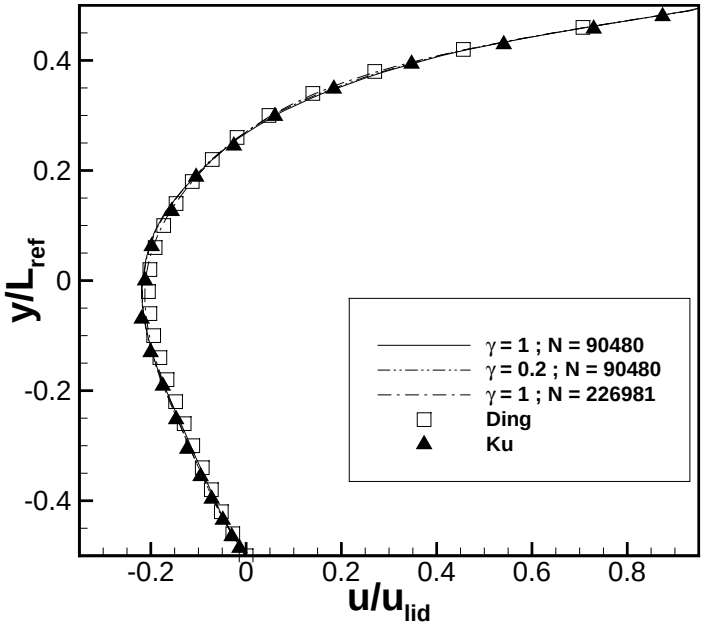
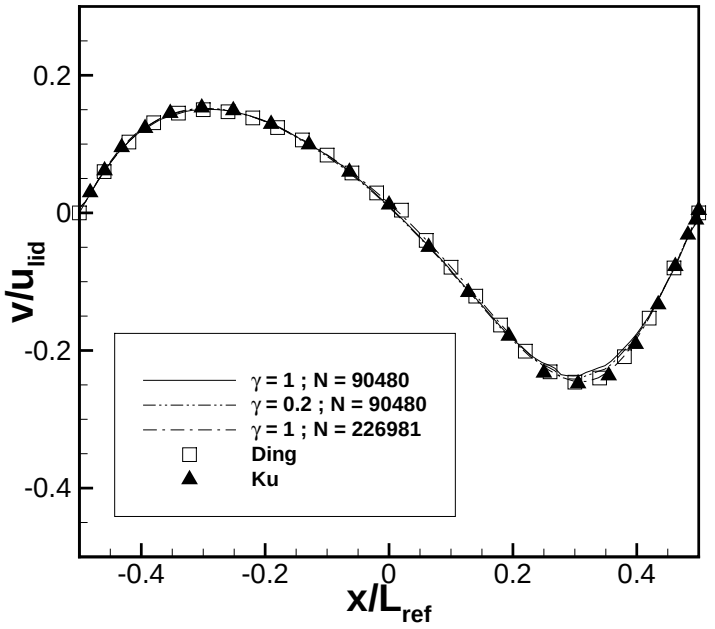


FIGURE 4.3: 3D lid-driven cavity unstructured mesh (90480 cells) and problem setup.

numbers. The reduction in iterations required to reach convergence compared to the non-preconditioned case is 9.72x at $Re = 1000$ and 5.62x at $Re = 100$ for $Ma = 0.017$. At the higher Mach number of $Ma = 0.17$, the decrease in CN is offset by disproportionately larger increase in $\Gamma_{vc,max}$ (see Table 4.2). This accounts for the lack of significant acceleration in convergence at the higher Mach number. An alternative to preconditioning to improve accuracy is to increase the mesh density. Figures 4.4 to 4.7 show that the preconditioned test cases produce results that are of similar accuracy to those of the unpreconditioned test cases on finer meshes. This is also with the benefit of a reduction in the iterations required to reach convergence and a reduction in the flux calculations per iteration. As can be seen in Figure 4.9 and Figure 4.10, this effect is most prominent for high Re and low Ma. The preconditioned case of $Re = 1000$ and $Ma = 0.017$ uses 5.87x less cells and 20.67x less iterations than the unpreconditioned case on the finest mesh. In this case preconditioning enables a 121x reduction in computational effort while attaining similar levels of accuracy.

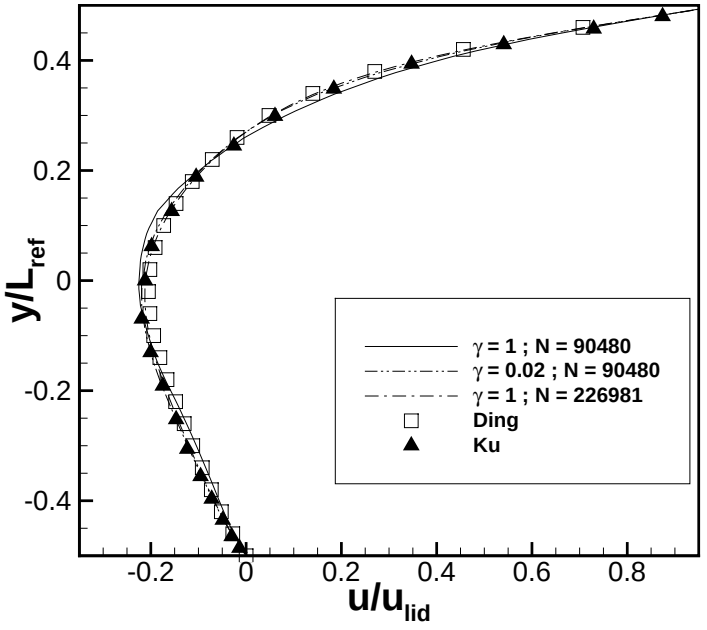


(a)

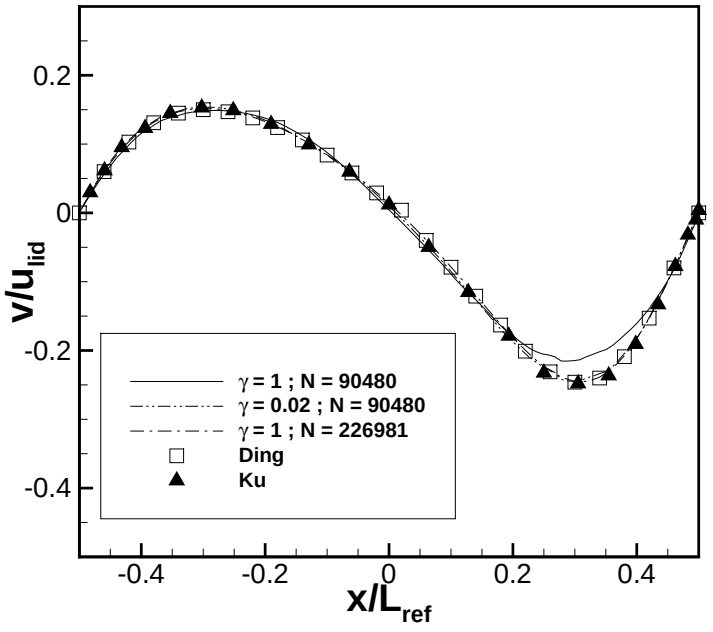


(b)

FIGURE 4.4: 3D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 100$, $Ma = 0.17$ and $u_{lid} = 0.1$.

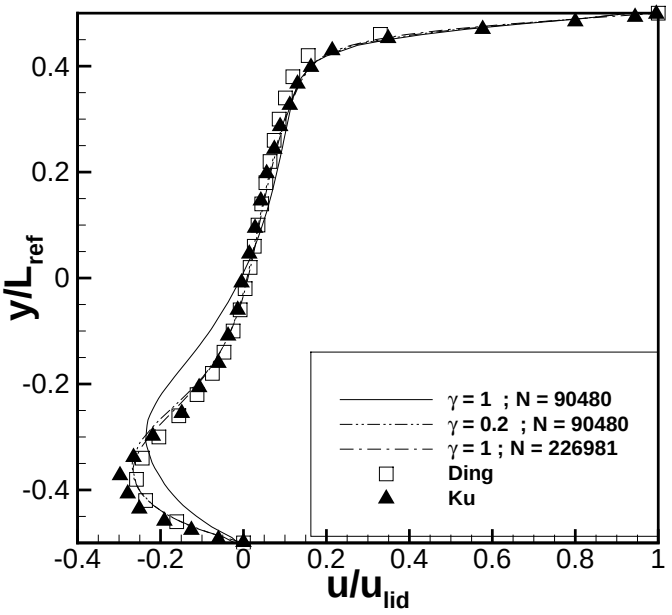


(a)

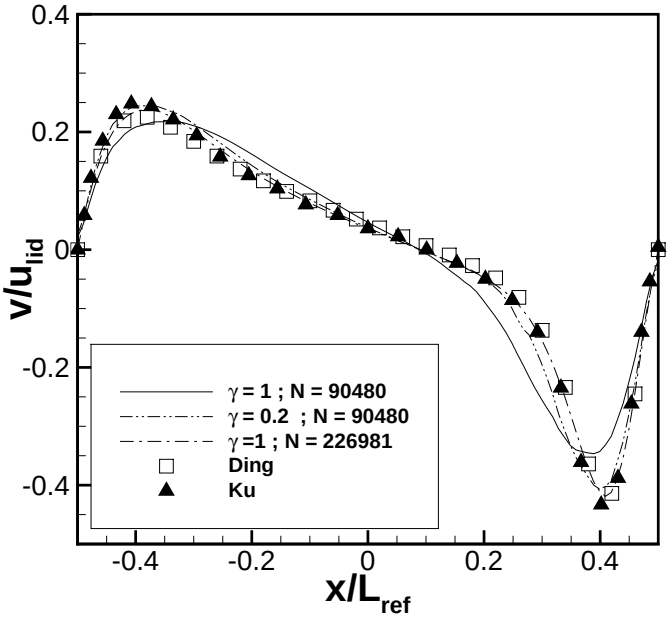


(b)

FIGURE 4.5: 3D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 100$, $Ma = 0.017$ and $u_{lid} = 0.01$.

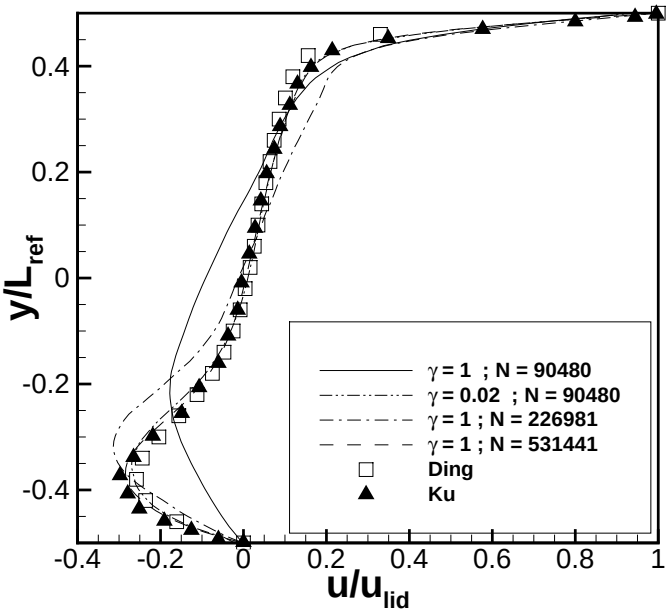


(a)

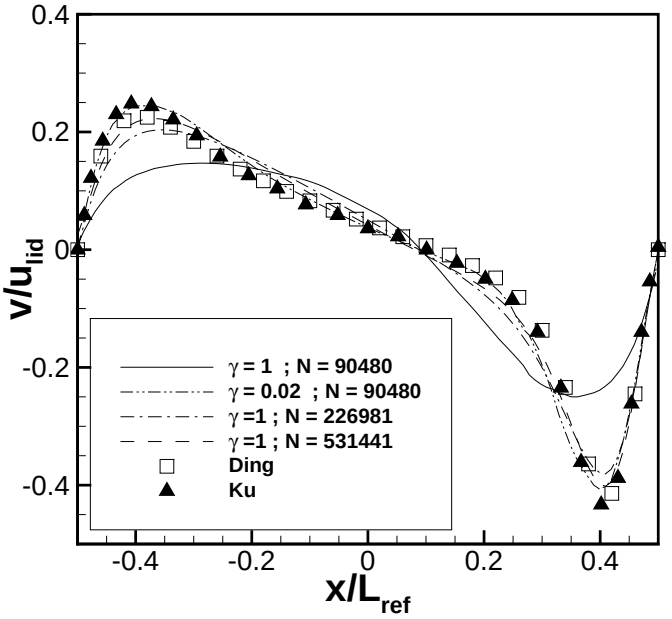


(b)

FIGURE 4.6: 3D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 1000$, $Ma = 0.17$ and $u_{lid} = 0.1$.



(a)



(b)

FIGURE 4.7: 3D lid-driven cavity flow: normalised a) u and b) v velocity profiles along vertical and horizontal centrelines respectively in the $z = 0$ plane for $Re = 1000$, $Ma = 0.017$ and $u_{lid} = 0.01$.

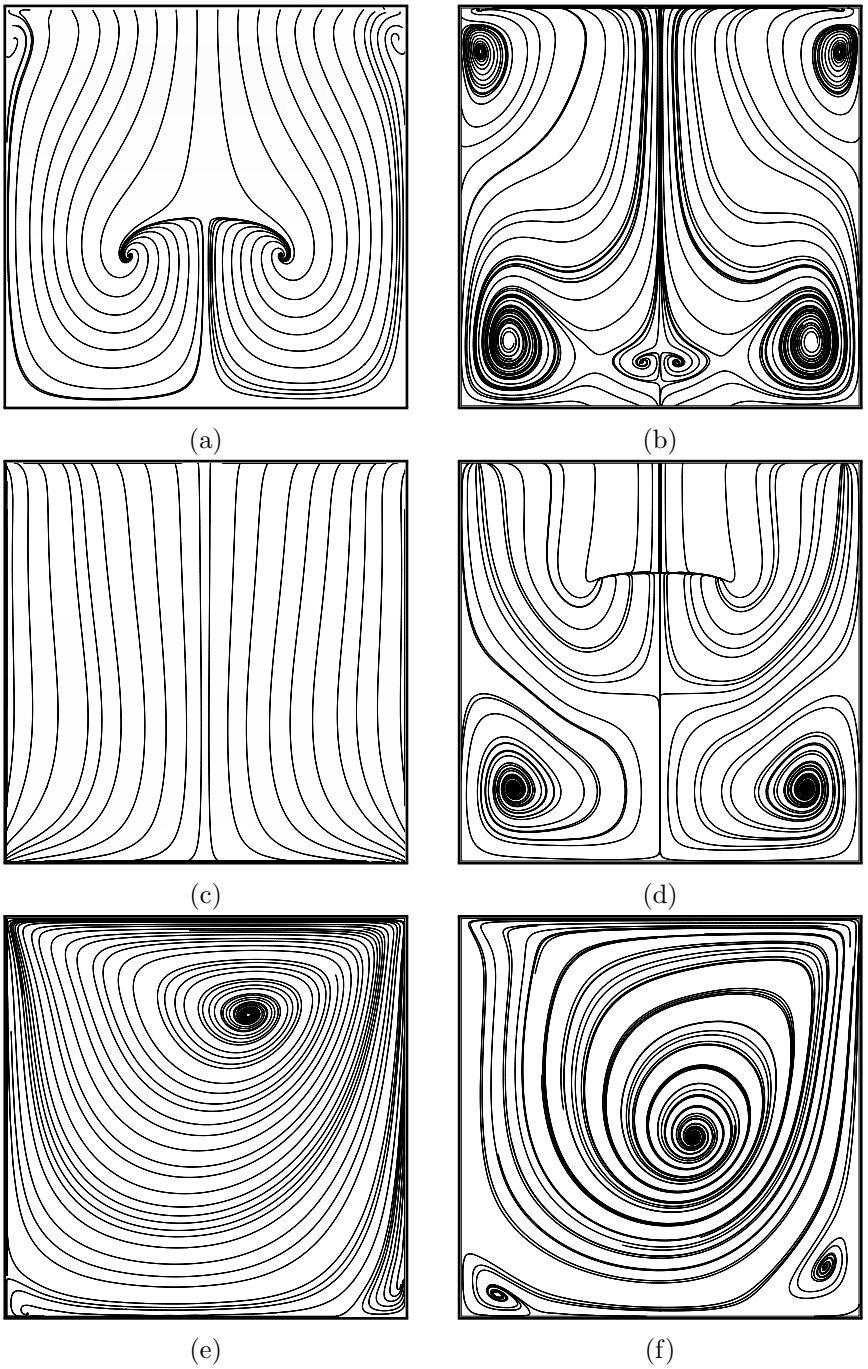
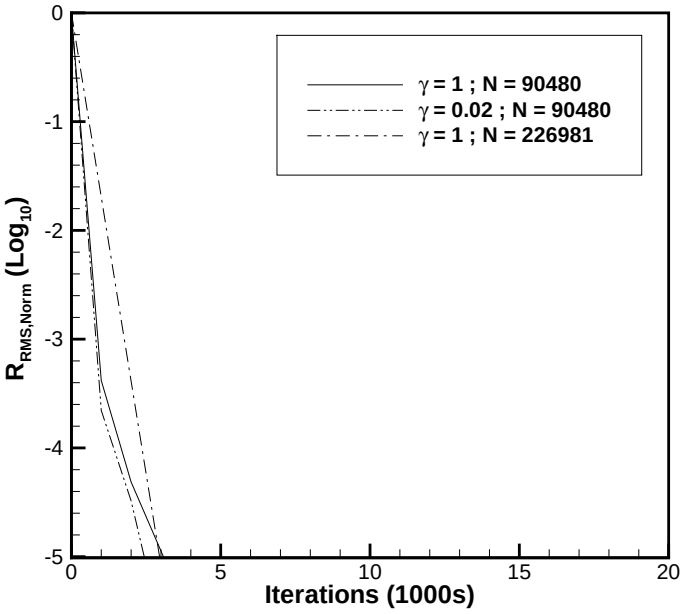
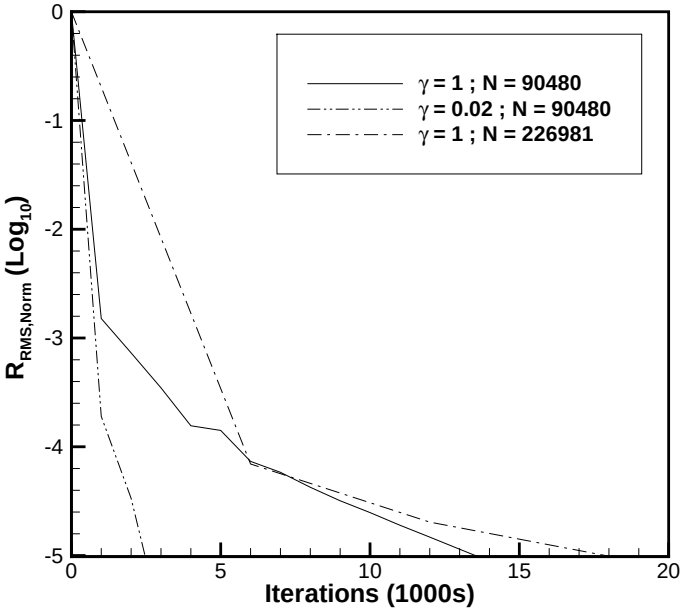


FIGURE 4.8: 3D lid-driven cavity flow: predicted streamlines in the $x = 0$ plane for a) $Re = 100$ and b) $Re = 1000$, in the $y = 0$ plane for c) $Re = 100$ and d) $Re = 1000$ and in the $z = 0$ plane for e) $Re = 100$ and f) $Re = 1000$.

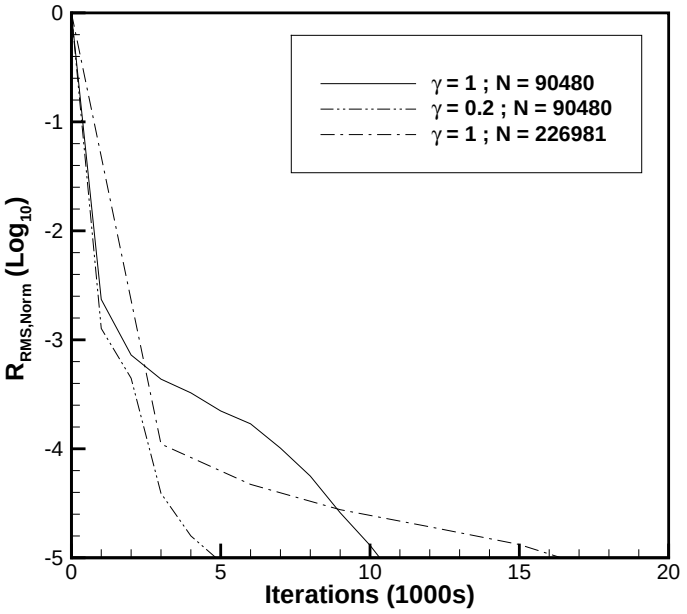


(a)

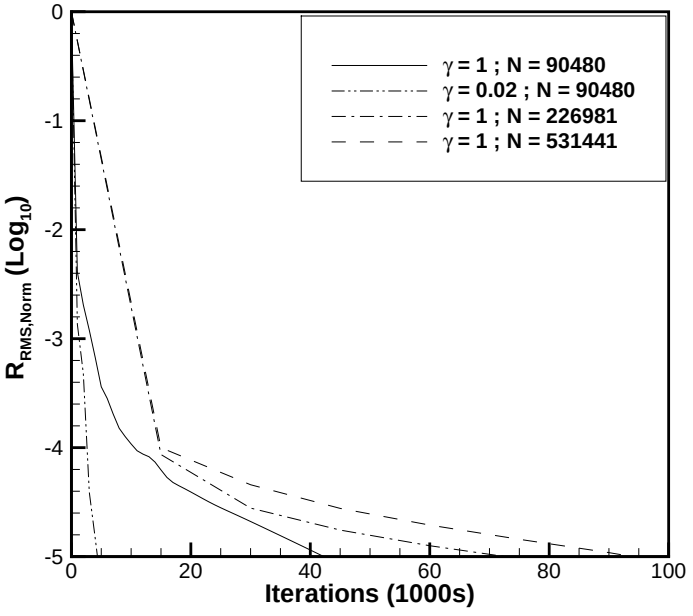


(b)

FIGURE 4.9: 3D lid-driven cavity flow: convergence history for $Re = 100$ with a) $u_{lid} = 0.1$ and b) $u_{lid} = 0.01$.



(a)



(b)

FIGURE 4.10: 3D lid-driven cavity flow: convergence history for $Re = 1000$ with a) $u_{lid} = 0.1$ and b) $u_{lid} = 0.01$.

4.4.3 3D Flow Over a Circular Cylinder

As mentioned previously, one of the motivations behind using the LBFS is to avoid the staircase approximation of curved boundaries with the traditional LBM. To demonstrate the applicability of the LBFS to flow problems with curved boundaries, 3D flow over a circular cylinder was investigated. This is a steady flow for low Reynolds numbers and so the PLBFS can also be used to solve this flow problem. The problem was set up as in Figure 4.11, with the cylinder immersed in a uniform freestream. The diameter of the cylinder was the reference length L_{ref} with a value of $L_{ref} = 1$ and spanned the length of the z-domain. The boundary condition at the inlet boundary was set equal to the freestream density ρ_∞ and freestream velocity U_∞ using Dirichlet boundary conditions. At the outlet boundary condition the pressure was set equal to static pressure by a Dirichlet boundary condition and a zero change in velocity at the outlet was maintained by a Neumann boundary condition. A no-slip boundary condition was applied to the wall of the cylinder and the remaining boundaries all have a zero change in velocity and density which are maintained by Neumann boundary conditions. The initial conditions were set as $\mathbf{V} = 0$ and $\rho = 1$. The flow problem was run for different combinations of Re, Ma and γ. The test cases parameters are summarised in Table 4.3. The values of γ were chosen for each test case as per the approach given by Table 4.1.

Test Case No.	U_∞	Re	Ma	γ	CN	$\Gamma_{vc,max}$	δx_{max}
1	0.1	20	0.17	1	6.77	4.41	0.173
2	0.1	20	0.17	0.2	2.15	22.05	0.866
3	0.01	20	0.017	1	58.73	0.44	0.017
4	0.01	20	0.017	0.05	3.88	8.82	0.346
5	0.1	40	0.17	1	6.77	2.20	0.086
6	0.1	40	0.17	0.2	2.15	11.04	0.433
7	0.01	40	0.017	1	58.73	0.22	0.008
8	0.01	40	0.017	0.05	3.88	4.41	0.173

TABLE 4.3: Individual test case parameters for 3D flow over a circular cylinder.

All test cases were run with a variety of meshes with densities of 7021, 13051, 18517, 23073 and 25877 cells. For each mesh, the cell height of the first cell

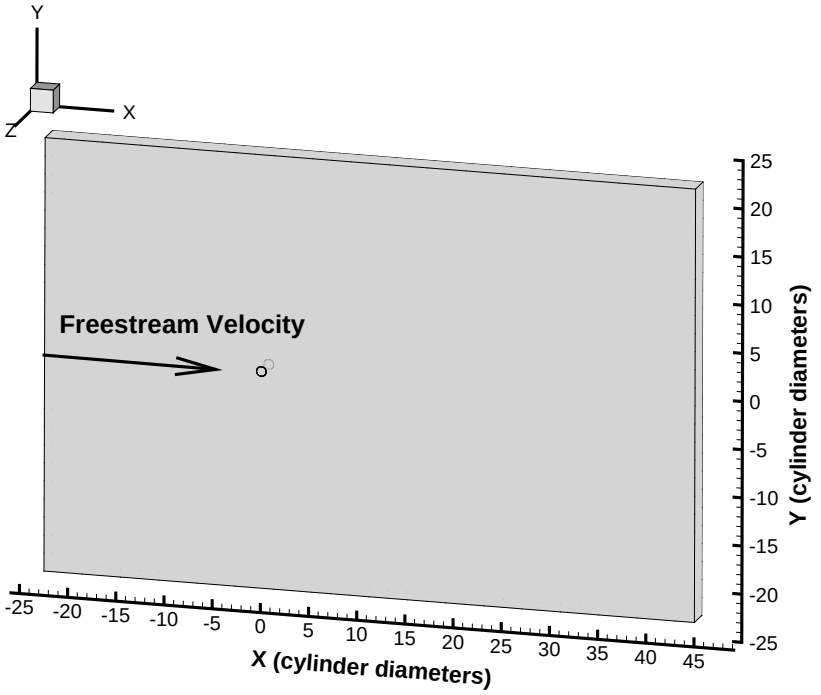


FIGURE 4.11: 3D flow over a cylinder problem setup.

adjacent to the cylinder wall was chosen as 0.05 and was one cell thick in the z-direction. The mesh containing 23073 cells is shown in Figure 4.12. For $Re = 20$ and $Re = 40$, the flow is steady. The predicted streamlines and velocity contours for Test Case 1 and 5 are shown in Figure 4.13. The streamtraces show stationary recirculation regions on the lee side of the cylinder. The flow pattern shown is consistent with the existing studies in the literature; however more detailed measures are required to demonstrate the performance of the PLBFS. To do this, a variety of parameters can be employed including the drag coefficient of the cylinder, the recirculation length and separation angle. The drag coefficient C_d relates the force acting on a body by a fluid in the direction of the freestream velocity to the force of the freestream dynamic pressure acting on the frontal area of the body. This is calculated as follows:

$$C_d = \frac{\gamma F_d}{0.5\rho_\infty U_\infty^2 A_f} \tag{4.22}$$

where A_f is the frontal area (the area projected onto a plane normal to the flow direction) of the body and F_d is the drag force which in this flow problem is calculated by integrating the pressure and viscous stresses acting in the x-direction

over the surface of the cylinder, *i.e.*:

$$F_d = -\int_S \left(\left[\begin{array}{ccc} \overline{p} - \mu\left(2\frac{\partial u}{\partial x}\right) & -\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & -\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \end{array} \right] \cdot \mathbf{n} \right) dS \qquad (4.23)$$

The pressure and viscous stresses in Equation (4.23) were calculated using Equation (4.1). The recirculation length L_s is defined as the distance between the trailing edge of the cylinder and the stagnation point in the wake *i.e.* where $\mathbf{V} = 0$ on the x-axis. The separation angle θ_s is the angle between the trailing edge and the point where the boundary layer separates from the cylinder. This point is indicated by the wall shear stress equalling zero.

To establish that the PLBFS is at least second-order accurate, Test Case 5 has also been performed for a variety of mesh densities where the first cell height, number of divisions and maximum element length were scaled in the same manner from $0.2L_{ref}$ to $0.02L_{ref}$. A further test case was performed at a scale of $0.01L_{ref}$ and this was used as the benchmark drag coefficient. The percentage error in the drag coefficient was calculated for each test case and is plotted in logarithmic form in Figure 4.14. This shows that the PLBFS is at least second order accurate which is consistent with the work of Shu et al. [137].

The results from the PLBFS are compared to previous numerical studies (Shu et al. [137], Dennis and Chang [190], Shukla et al. [191] and Pellerin et al. [138]) in Table 4.4 and Table 4.5. The variation of the various parameters with mesh density are also plotted and compared to the upperbound and lowerbound values from the literature in Figures 4.15 to 4.17. These results show that for the vast majority of test cases, the parameters are converged or very near convergence on a mesh of 23073 cells. The main differentiator in accuracy are the parameters $\Gamma_{vc,max}$ and δx_{max}. The most accurate predictions are for those test cases which have reduced values of $\Gamma_{vc,max}$ while having δx_{max} greater than the first cell height in the boundary layer. These parameters are dependent on the choice of Ma and γ and a balance has to be struck between both to ensure maximum accuracy.

The convergence histories are plotted in Figure 4.18. As shown in Figure 4.18, the reduction in iterations required to reach convergence compared to the non-preconditioned case is 4.92x at $Re = 20$ and 9.625x at $Re = 40$ for $Ma = 0.017$. However there was an increase in iterations at the higher Mach number of 0.17. Due to the detrimental effect on convergence rates at $Ma = 0.17$, further test cases were run to investigate this issue. These additional test cases were chosen

Reference	U_∞	γ	C_d	L_s/L_{ref}	θ_s
Shu et al. [137]	0.1	-	2.06	0.94	42.94
Dennis and Chang [190]	-	-	2.05	0.94	43.70
Shukla et al. [191]	-	-	2.07	0.92	43.30
Pellerin et al.[138]	-	-	2.01	0.92	43.58
Test Case 1	0.1	1	2.05	0.91	42.97
Test Case 2	0.1	0.2	2.1	0.98	42.81
Test Case 3	0.01	1	2.07	0.86	40.86
Test Case 4	0.01	0.02	2.04	0.91	42.97

TABLE 4.4: 3D flow over a circular cylinder: comparison of predicted drag coefficient, recirculation length and separation angle with predictions in the literature for Re =20 and N_{cells} = 23073.

Reference	U_∞	γ	C_d	L_s/L_{ref}	θ_s
Shu et al. [137]	-	-	1.53	2.24	52.69
Dennis and Chang [190]	-	-	1.52	2.35	53.80
Shukla et al. [191]	-	-	1.55	2.34	52.70
Pellerin et al. [138]	-	-	1.5	2.26	53.52
Test Case 5	0.1	1	1.53	2.23	53.13
Test Case 6	0.1	0.2	1.58	2.4	52.64
Test Case 7	0.01	1	1.54	2.00	49.26
Test Case 8	0.01	0.02	1.53	2.21	52.64

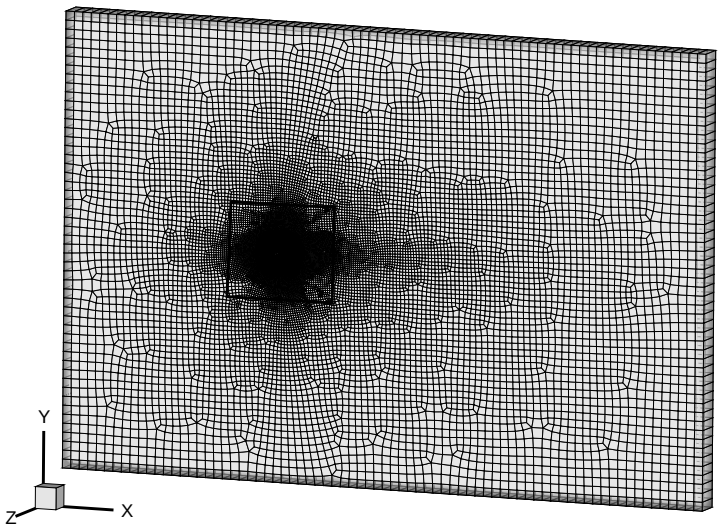
TABLE 4.5: 3D flow over a circular cylinder: comparison of predicted drag coefficient, recirculation length and separation angle with predictions in the literature for Re =40 and N_{cells} = 23073.

as per Table 4.1 and are shown in Table 4.6. Test Cases 9-12 used the same mesh with N_{cells} = 23073 as before but with a more moderate level of preconditioning. The convergence histories of these additional test cases can be seen in Figure 4.19. These results show that any level of preconditioning has an adverse effect on convergence rates for $Ma = 0.17$.

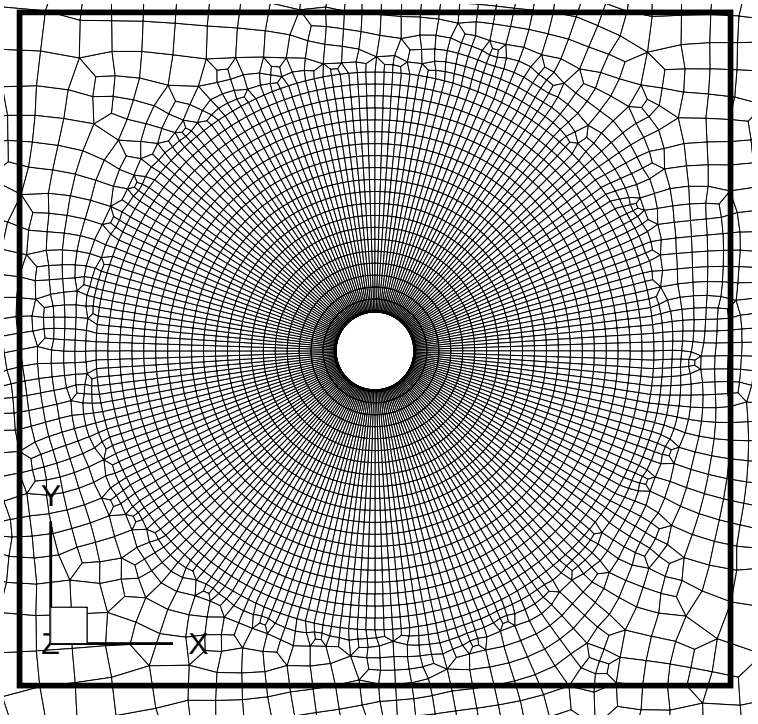
Test Case No.	U_∞	Re	Ma	γ	CN	$\Gamma_{vc,max}$	δx_{max}
9	0.1	20	0.17	0.5	3.88	8.21	0.346
10	0.1	20	0.17	0.75	5.33	5.88	0.230
11	0.1	40	0.17	0.5	3.88	4.41	0.173
12	0.1	40	0.17	0.75	5.33	2.94	0.115

TABLE 4.6: Additional individual test case parameters for 3D flow over a circular cylinder problem.

Using an unstructured hexahedral mesh produces excellent results that agree with existing studies in the literature. The previous studies using the LBFS were performed by Shu et al. [137] with a cell-centred non-uniform O-grid mesh and by Pellerin et al. [138] with a vertex-centred tetrahedral mesh. They achieved excellent accuracy using 60501 cells and 63541 vertices respectively. Using the PLBFS on a fully unstructured hexahedral grid allowed similar levels of accuracy to be attained while using a mesh with 23073 cells. Inspecting Table 4.3 shows that preconditioning allowed the use of larger δx_{max} in larger cells in the mesh. This lead to more accurate results. It can be seen that Test Cases 3 and 7, which do not employ preconditioning, have δx_{max} values that are an order of magnitude lower than the preconditioned cases. In both cases, δx_{max} was less than the cell height of the first cell in the boundary layer. This results in drag coefficients, recirculation lengths and separation angles which were not as accurate as in the preconditioned cases. From the initial results in Figure 4.18, preconditioning had a significant acceleration effect on convergence at $Ma = 0.017$ but had a detrimental effect on the rate of convergence for $Ma = 0.17$. This can be attributed to the fact that the vast majority of the cells in the mesh can be considered viscous dominated for $\gamma = 1$ as the Reynolds number was very low in each test case. Any implementation of preconditioning for $Ma = 0.17$ would make these cells increasingly viscous dominated and increase runtimes.

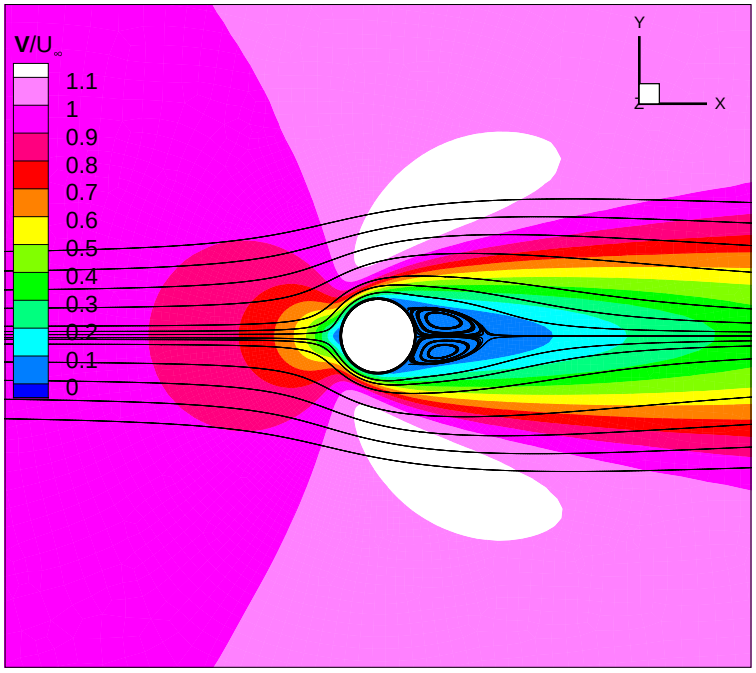


(a)

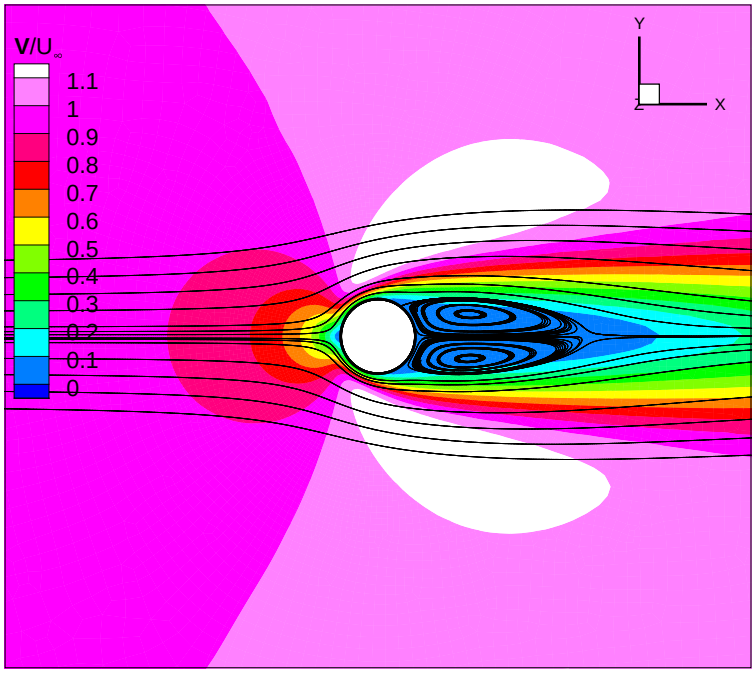


(b)

FIGURE 4.12: 3D flow over a circular cylinder: a) unstructured hexahedral mesh with 23073 cells and b) exploded view of the mesh close to cylinder surface.



(a)



(b)

FIGURE 4.13: 3D flow over a circular cylinder: streamlines and velocity contour plots on the $z = 0$ plane for a) $Re = 20$ and b) $Re = 40$, $Ma = 0.17$, γ =1 and $N_{cells} = 23073$.

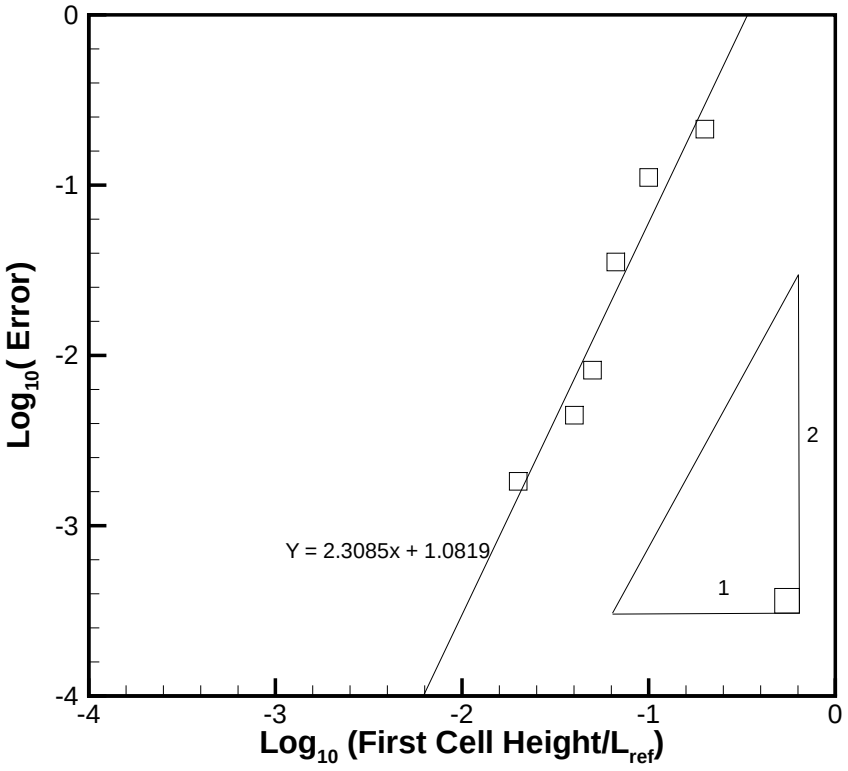
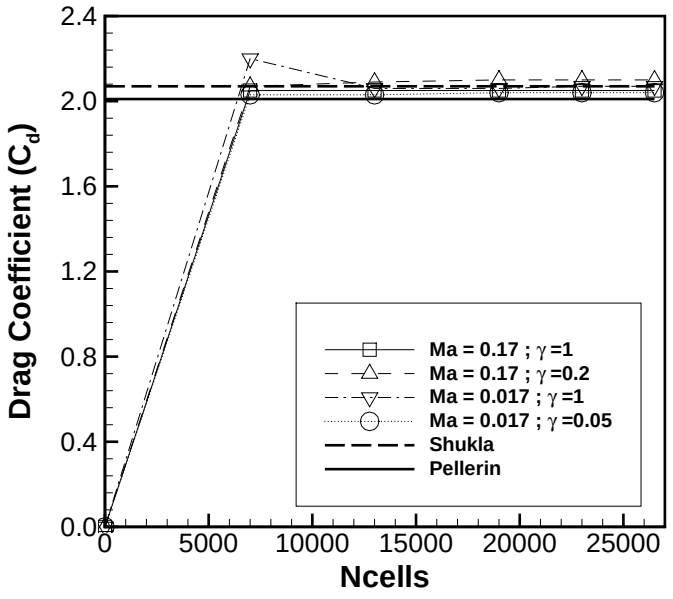
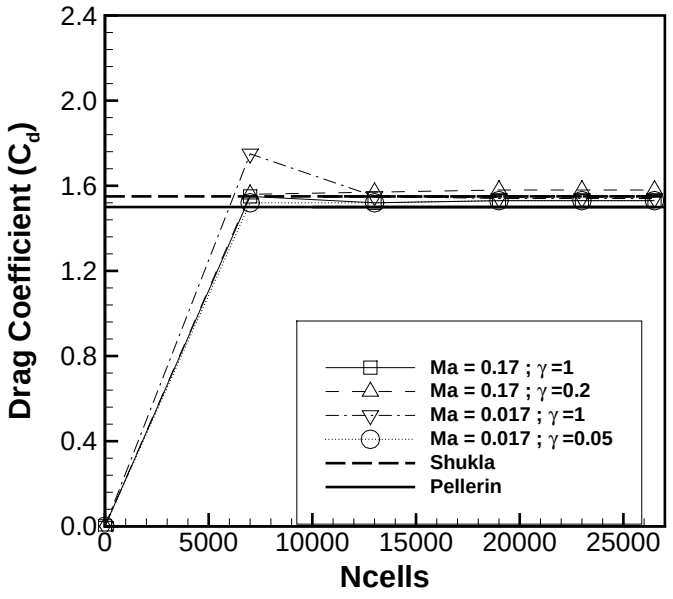


FIGURE 4.14: Variation in percentage error in drag coefficient with mesh size. The percentage error is relative to a simulation with first cell height on the cylinder boundary equal $0.01L_{ref}$. All simulations were run for $Re = 40$, $Ma = 0.17$ and γ =1.

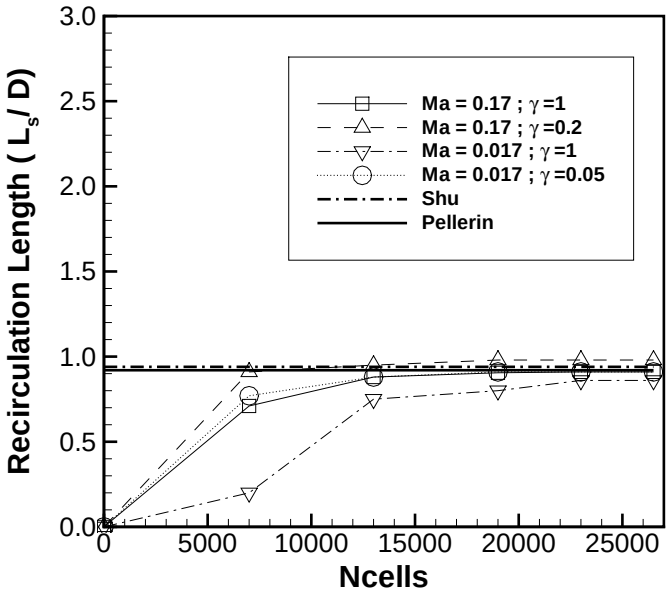


(a)

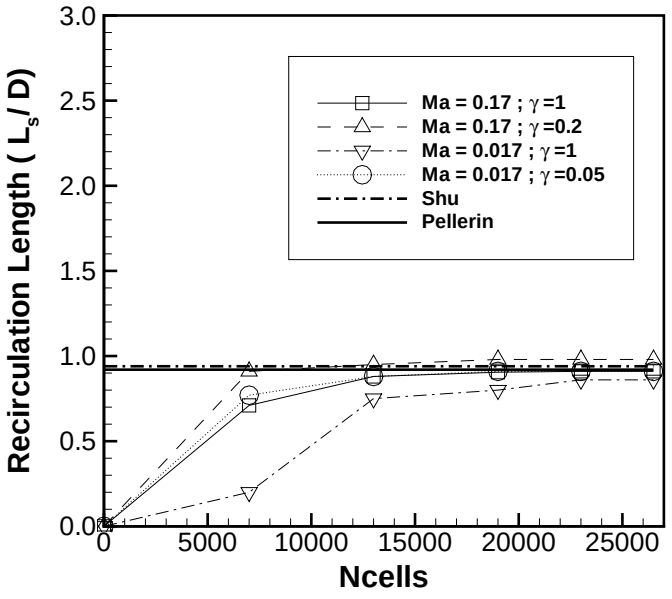


(b)

FIGURE 4.15: 3D flow over a circular cylinder: variation of drag coefficient with mesh density for varying U_∞ and γ for a) $Re = 20$ and b) $Re = 40$.

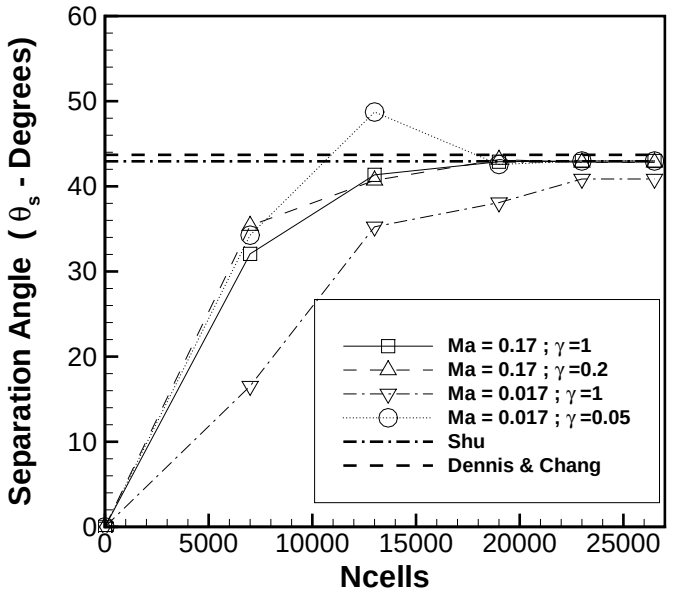


(a)

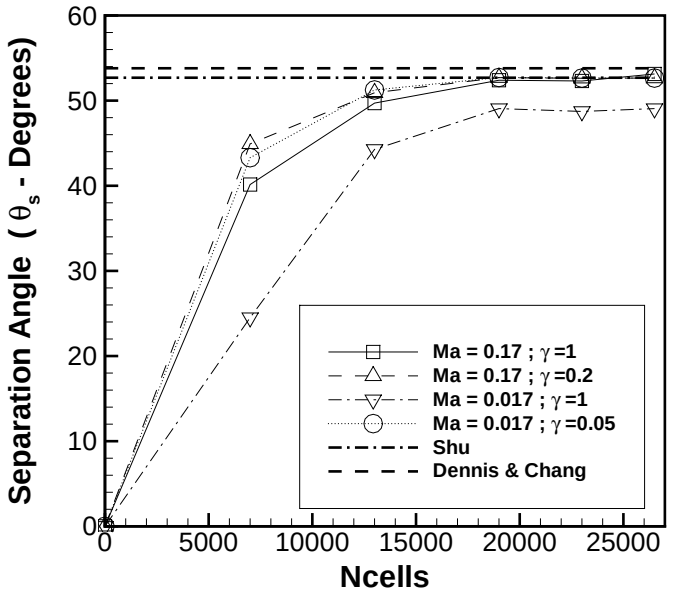


(b)

FIGURE 4.16: 3D flow over a circular cylinder: variation of recirculation length with mesh density for varying U_∞ and γ for a) $Re = 20$ and b) $Re = 40$.

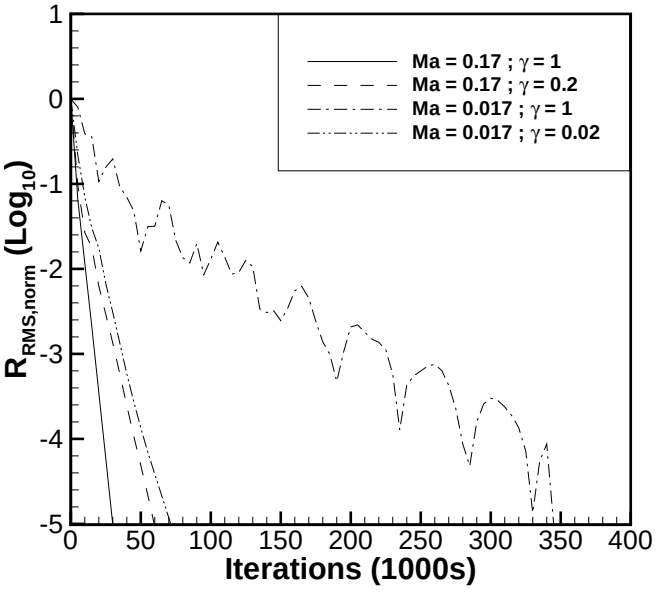


(a)

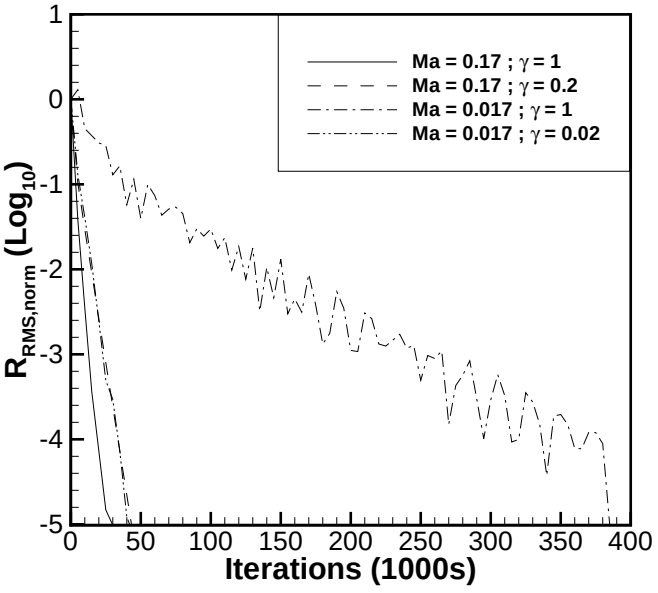


(b)

FIGURE 4.17: 3D flow over a circular cylinder: variation of separation angle with mesh density for varying U_∞ and γ for a) $Re = 20$ and b) $Re = 40$.

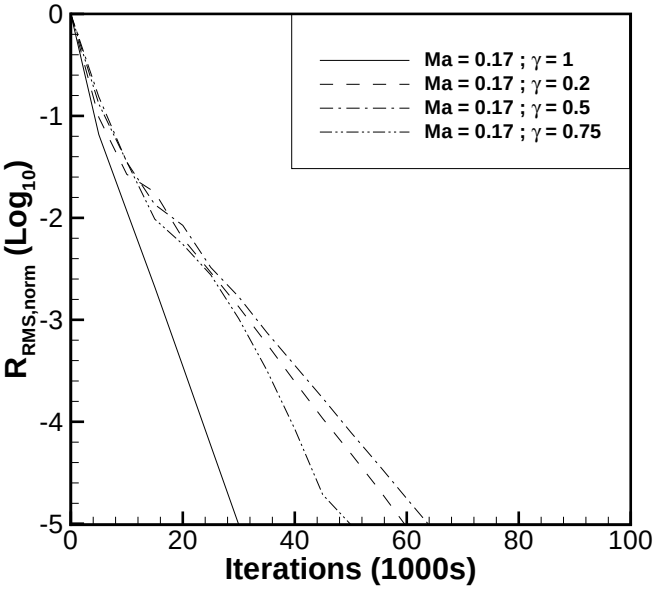


(a)

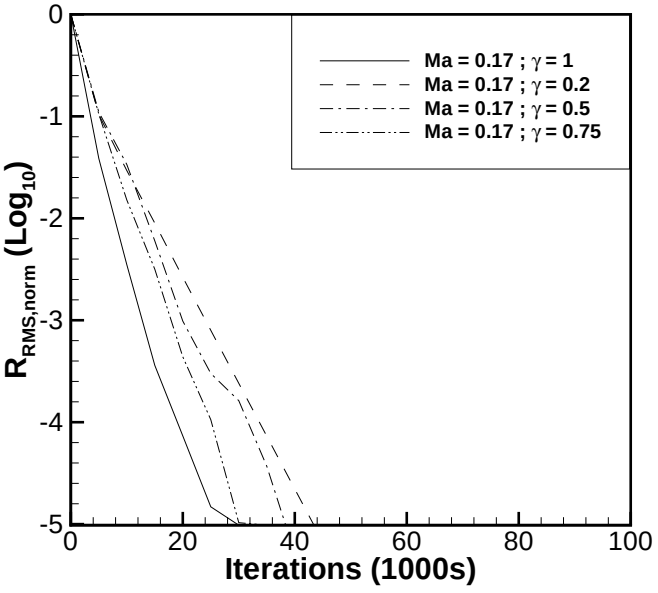


(b)

FIGURE 4.18: 3D flow over a circular cylinder: convergence history for a) $Re = 20$ and b) $Re = 40$ on a mesh with $N_{cells} = 23073$.



(a)



(b)

FIGURE 4.19: 3D flow over a circular cylinder: convergence history for a) $Re = 20$ and b) $Re = 40$ for a range of γ on a mesh with $N_{cells} = 23073$.

4.5 Summary

In this chapter a PLBFS has been successfully used to solve flow problems on 3D unstructured hexahedral meshes. This was demonstrated by successfully simulating 3D lid-driven cavity flow and 3D flow over a circular cylinder. The use of unstructured hexahedral meshing was shown to enable the use of coarser meshes than structured meshes with similar levels of accuracy. This accuracy was also further increased by using preconditioning as it allows the use of a more optimal local lattice Boltzmann reconstruction in larger cells. This effect was demonstrated to be more significant at higher Reynolds numbers and lower Mach numbers. Preconditioning also has the benefit of accelerating convergence in the vast majority of cases. Again this effect was more beneficial at higher Reynolds numbers and lower Mach numbers as the unpreconditioned condition number is larger. It has also been shown that in certain cases, such as flow problems with low Reynolds numbers and very fine mesh refinement in the boundary layer as with flow over a circular cylinder, the acceleration due to preconditioning can be minimal. In such cases it is recommended to use moderate levels of preconditioning, so that the ratio of the viscous spectral radius to the convective spectral radius remains less than one or to use a coarser grid where possible if sufficient accuracy can be achieved with such a mesh. The PLBFS has been shown to greatly enhance the use of unstructured meshes with the LBFS, particularly in flow problems with higher Reynolds numbers. It has also been shown that the 3D PLBFS is at least second-order accurate in space for the flow over a cylinder problem.

Chapter 5

Immersed Boundary Method

5.1 Introduction

As discussed in Section 1.6, a method for modelling FSI between RBCs and plasma is required to model blood flow. The IBM was the method that was identified as the best choice for modelling this FSI between RBC and plasma. In this chapter the IBM and its variants are introduced. The extensions of the IBM to be compatible with unstructured and non-uniform grids are then discussed. The results of two benchmark flow problems that contain FSI are then shown to demonstrate the efficacy of the LBFS and IBM in modelling FSI flow problems.

5.2 Overview of Immersed Boundary Method

The IBM is a method that can be used to solve FSI problems. It involves immersing a Lagrangian object within a fluid domain and approximating a boundary of the object via a force transfer between nodes on the Lagrangian object and the cells within a discretised fluid domain. It involves the following main steps:

- Interpolation of fluid velocities onto Lagrangian nodes.
- Integration in time of Lagrangian node velocities to calculate node position.
- Calculation of force at each Lagrangian node based off new position.
- Spreading of node forces to discretised cells in the fluid.

- Solving the N-S equations at each cell using body forces from Lagrangian nodes.

There are many variations of the IBM but they all follow the above high-level process. The IBM was originally introduced by Peskin [101] for solving FSI problems with deformable objects. The majority of benchmark FSI flow problems for verifying code involve rigid objects which don't deform. Many flavours of the IBM have been suggested which capture the no-slip boundary conditions of rigid objects. These include the explicit feedback IBM for rigid-boundaries first suggested by Lai and Peskin [192] and introduced for the LBM by Feng and Michaelides [193]. This method involves attaching the Lagrangian nodes to their reference position by a spring and any displacements from the reference position result in a restoring force. If a suitable stiffness of the spring is chosen, this can correctly model rigid objects. The stiffness of the spring is a free parameter that can impact on the stability and accuracy of the method.

Due to this weakness, much effort has been put into IBM methods without a free parameter. This family of methods are referred to as Direct-Forcing-IBM (DF-IBM) methods. An explicit DF-IBM was first proposed for the FVM by Kim [194] and for the LBM by Feng and Michaelides [195]. This family of methods involves calculating a corrective force at the Lagrangian node that is equivalent to the velocity correction required to make the velocity at the node equal to the known boundary velocity. An implicit method was proposed by Wu and Shu [196] that involves calculating the unknown force densities required to ensure the correct boundary velocities on the Lagrangian object. This approach requires matrix inversions for calculating the force densities. This results in a method that probably offers the most accurate IBM approach for ensuring no-slip boundary conditions but comes at the cost of a very expensive matrix inversion where a large number of Lagrangian nodes are used. It has also been extended to the LBFS which is the N-S solver used in this work [197, 198]. Further efforts have also been made to avoid the expensive matrix inversion involved in the above method and have adopted an iterative solver to finding the unknown force densities. Khan and Hassan [199] provide a very detailed review of the different direct-forcing approaches in the literature.

In this work the original IBM of Peskin is adopted as the end goal is to model the deformation of red blood cells. To verify the implementation of the IBM, the

explict feedback IBM of Lai and Peskin is used on benchmark flow problems with rigid objects. This includes flow over a cylinder and flow over a sphere. This tested all aspects of the IBM for deformable objects with the exception of force calculations at the Lagrangian nodes. This step was replaced by the constitutive equations of motion of the RBC at a later stage.

5.3 Governing Equations and Implementation

5.3.1 Lagrangian System

The IBM consists of a fixed Eulerian mesh and a deformable Lagrangian surface. The fixed Eulerian mesh (coordinates $\mathbf{r}$) is used to discretise the fluid computational domain for solving the N-S equations. The Lagrangian surface of an immersed object is discretised into nodes and these nodes are free to move in space (coordinates $\mathbf{R}$). An example of a Eulerian/Lagrangian setup is provided in Figure 5.1.

Note that the Lagrangian nodes are not connected but free to move individually. Connectivity relations may be required depending on the constitutive equations of motion of the object.

5.3.2 Continuous Governing Equations

The first key principle of the IBM is that the velocity of the Lagrangian nodes is equal to the fluid velocity $\mathbf{V}$ at the same coordinates, *i.e*:

$$\frac{d\mathbf{R}(t)}{dt} = \mathbf{V}\left(\mathbf{r}, t\right) \tag{5.1}$$

Rewriting the right hand side in terms of Eulerian coordinates gives:

$$\frac{d\mathbf{R}(t)}{dt} = \int d^3r \mathbf{V}(\mathbf{r}, t)\delta(\mathbf{r} - \mathbf{R}(t)) \tag{5.2}$$

where $\delta(\mathbf{r} - \mathbf{R}(t))$ is the Dirac delta distribution. The second key principle of the IBM describes the momentum exchange between the Lagrangian nodes and the Eulerian mesh. Assuming there is a force density (per unit area) $\mathbf{F}_a$ acting on the

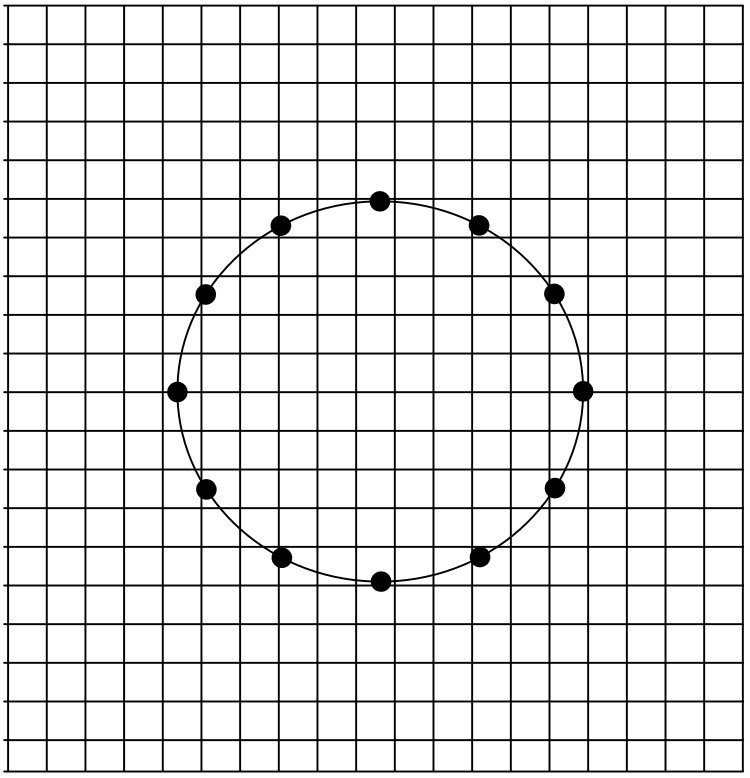


FIGURE 5.1: A Lagrangian mesh defining a cylinder overlaying an Eulerian mesh. Lagrangian nodes are defined by black dots.

Lagrangian nodes, then the equivalent force density (per unit volume) acting on the Eulerian mesh is given by:

$$\mathbf{F}_{vol}(\mathbf{r},t) = \int d^2R \mathbf{F}_a(\mathbf{R}(t),t)\delta(\mathbf{r}-\mathbf{R}(t)) \tag{5.3}$$

This equation describes the spreading of the Lagrangian forces to the Eulerian mesh and is called *force spreading*.

5.3.3 Discretised Governing Equations

The next step is to discretise Equation (5.1) and Equation (5.3). Peskin [101] provides a detailed derivation which results in the final discretised IBM equations:

$$\frac{d\mathbf{R}_j(t)}{dt} = \sum_{\mathbf{r}\in g_{j,h}} \mathbf{V}(\mathbf{r},t)\delta_h(\mathbf{r}-\mathbf{R}_j(t))\Delta x^3 \tag{5.4}$$

and

$$\mathbf{F}_{vol}(\mathbf{r},t) = \sum_{j \in G_{\mathbf{r},h}} \mathbf{F}_{a,j}(\mathbf{R},t)\delta_h(\mathbf{r} - \mathbf{R}_j(t))\Delta A \tag{5.5}$$

where j is the j^{th} Lagrangian node in the immersed object, h is the size of the stencil of the Dirac delta distribution, $\mathbf{R}_j$ is the position of the j^{th} node, Δx is the Cartesian edge length of a uniform cell in the Eulerian mesh, $g_{j,h}$ are the Eulerian cells that lie within the stencil of the Dirac delta distribution, $G_{\mathbf{r},h}$ are the Lagrangian nodes that lie in the stencil of a cell $\mathbf{r}$, $\mathbf{F}_{a,j}$ is the force density per unit area at Lagrangian node j and ΔA is the surface area associated with Lagrangian node j.

The next step is to adopt a suitable discretised version of the Dirac delta distribution function. Peskin derived suitable kernel functions to represent δ_h using the following assumptions. First it is assumed that the 3D δ_h can be represented by a product of one-dimensional (1D) variables, *i.e.* kernels, that scale with the edge length Δx as follows:

$$\delta_h = \frac{\phi_h(x)\phi_h(y)\phi_h(z)}{\Delta x^3} \tag{5.6}$$

The following assumptions and restrictions are used when creating the 1D variables in Equation (5.6):

- Interpolation and spreading should be short-ranged. This is required to reduce computational overhead as much as possible.
- Momentum and angular momentum have to be identical when evaluated either in the Eulerian or the Lagrangian system (same speed and rotation in both systems).
- Lattice artefacts ("bumpiness" of the interpolation when boundaries move) should be suppressed as much as possible.
- The kernel function has to be normalised: $\sum_r \Delta x^3 \delta_h(\mathbf{r}) = 1$

This results in the final formulation of the kernels as:

$$\phi_2(x) = \left\{ \begin{array}{c} 1 - |x| \\ 0 \end{array} \right\} \quad \begin{array}{l} \text{when } 0 \leq |x| \geq \Delta x \\ \text{when } \Delta x \leq |x| \end{array} \tag{5.7}$$

$$\phi_3(x) = \left\{ \begin{array}{c} \frac{1}{3}(1+\sqrt{1-3x^2}) \\ \frac{1}{6}(5-3|x|-\sqrt{-2+6|x|-3x^2}) \\ 0 \end{array} \right\} \begin{array}{l} \text{when } 0 \leq |x| \geq 0.5\Delta x \\ \text{when } 0.5\Delta x \leq |x| \geq 1.5\Delta x \\ \text{when } 1.5\Delta x \leq |x| \end{array} \tag{5.8}$$

$$\phi_4(x) = \left\{ \begin{array}{c} \frac{1}{8}(3-2|x|-\sqrt{1+4|x|-4x^2}) \\ \frac{1}{8}(5-2|x|-\sqrt{-7+12|x|-4x^2}) \\ 0 \end{array} \right\} \begin{array}{l} \text{when } 0 \leq |x| \geq \Delta x \\ \text{when } \Delta x \leq |x| \geq 2\Delta x \\ \text{when } 2\Delta x \leq |x| \end{array} \tag{5.9}$$

The index on the ϕ function denotes the number of cells included in the stencil of the kernel in each Cartesian direction. This results in 2^3, 3^3 and 4^3 calculations per Lagrangian node for force spreading and velocity interpolation. A visual representation of the kernels stencils is provided in Figure 5.2 :

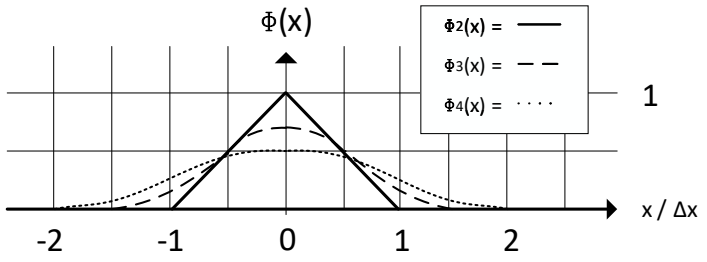


FIGURE 5.2: Various stencils for calculating the Dirac delta distribution function.

This then results in the following algorithm for the IBM:

- Interpolation of fluid velocities onto Lagrangian nodes using Equation (5.4), Equation (5.6) and Equations (5.7) to (5.9).
- Integration in time of Lagrangian node velocities to calculate change in node position.
- Calculation of force at each Lagrangian node based off new position. The force calculation is dependent on the mechanics of the object being modelled.
- Spreading of node forces to discretised cells in the fluid using Equation (5.5), Equation (5.6) and Equations (5.7) to (5.9).

- Solve the N-S equations at each cell using body forces from Lagrangian nodes.

5.4 Non-Uniform Grids

5.4.1 Introduction

It was discovered during the flow over a sphere benchmark case that in excess of 10^6 uniform cells were required to get sufficiently accurate results. This resulted in a flow problem with a computational effort that exceeded the hardware capacity that was available to the project at the time. Alternative approaches were investigated to reduce the computational effort. As the LBFS can be extended to use unstructured grids, a variety of IBMs applicable to non-uniform grids were investigated. The main difficulty with extending the IBM to unstructured/non-uniform grids is in the velocity interpolation and force spreading (VI-FS) calculations detailed earlier. The key aspect of these operations is that force and moments are conserved as per the following equations:

$$\sum \mathbf{F}_{vol}(\mathbf{r},t)\Delta V(\mathbf{r},t) = \sum_{j \in G_{\mathbf{r},h}} \mathbf{F}_{a,j}(\mathbf{R},t)\Delta A(\mathbf{R},t) \tag{5.10}$$

and

$$\sum \mathbf{r} \times \mathbf{F}_{vol}(\mathbf{r},t)\Delta V(\mathbf{r},t) = \sum_{j \in G_{\mathbf{r},h}} \mathbf{R} \times \mathbf{F}_{a,j}(\mathbf{R},t)\Delta A(\mathbf{R},t) \tag{5.11}$$

How various studies in the literature have satisfied the above equations on unstructured grids will now be discussed.

Ouro et al. [200] successfully used a moving least-squares formulation for VI-FS calculations on fully unstructured grids but this was for rigid stationary objects and required the calculation of n nearest neighbouring cells to the object nodes for VI-FS calculations. This becomes prohibitively expensive when using deformable moving objects and 3D meshes as the search per object node is of order *O(total cells in mesh)*. Pinelli et al. [201] use the Reproducing Kernel Particle Method developed by Liu [202] to calculate the VI-FS calculations. This approach involves the use of modified window functions to create a moment matrix which then requires a rescaling to avoid singularity. This approach can be applied to

unstructured and non-uniform grids alike. This approach was then successfully applied to flowing RBCs on fully unstructured grids by Mendez et al. [203]. The method of Pinelli was extended by Toja-Silva et al. [204] to use radial-basis functions to calculate the VI-FS calculations. This method again involves the creation of a moment matrix and the solving of a linear system of equations. It removes the rescaling of the moment matrix but includes an arbitrary parameter to describe the radial-basis functions. Finally Jang et al. [205] have used a reproducing polynomial particle method (RPPM) to calculate the VI-FS calculations on non-uniform grids. This method also generates a linear system of equations to ensure the conservation of moments in the system. However it doesn't involve any rescaling or arbitrary parameters. One massive advantage of the RPPM is that the moment matrix is in the form of a Vandermonde matrix and allows the linear system to be solved explicitly. For this reason the method of Jang was adopted for the IBM on non-uniform grids.

While all the above methods can be used on unstructured and non-uniform grids; it is proposed that only non-uniform meshes are used in this work due to their superior computational efficiency (see Section 8.5.1.3 for detailed analysis). In summary non-uniform grids allow neighbouring nodes to be found at a much cheaper computational cost then unstructured grids. If analytical functions are used to generate the grids, e.g. Chebyshev nodes, then the neighbouring nodes can be found explicitly. However if other functions are used, then the neighbouring nodes can be found using binary searches on each of the Cartesian axes. For a cube this results in order $O(3 \times log_2(number\ of\ x\ cells))$ which compares extremely favourably to a full search of the mesh which is of order $O((number\ of\ x\ cells)^3)$.

5.4.2 Governing Equations

Combining Equation (5.10) and Equation (5.5) gives the following condition per each object node:

$$\sum_{\mathbf{r} \in g_{j,h}} \delta_h(\mathbf{r} - \mathbf{R}_j(t)) \Delta V(\mathbf{r}, t) = 1 \tag{5.12}$$

To compensate for the non-uniformity of the cell volumes, a new corrected distribution is defined:

$$\delta_h^{'}(\mathbf{r} - \mathbf{R}_j(t)) = \frac{\delta_h(\mathbf{r} - \mathbf{R}_j(t))}{\Delta V(\mathbf{r}, t)} \tag{5.13}$$

which satisfies the zero-moment condition of the weight function:

$$\sum_{\mathbf{r}\in g_{j,h}} \delta_h^{'}(\mathbf{r}-\mathbf{R}_j(t)) = 1 \tag{5.14}$$

In three dimensions, the corrected weight function can be split into the product of its Cartesian terms as for the uniform case:

$$\delta_h^{'}(\mathbf{r}-\mathbf{R}_j(t)) = \phi^{'}(X_j - x)\phi^{'}(Y_j - y)\phi^{'}(Z_j - z) \tag{5.15}$$

where X_j , Y_j and Z_j are the cartesian scalars of the position of jth Lagrangian node. The weight functions can be obtained by solving the following systems of linear equations:

$$\begin{array}{l} \sum_{i=1}^{i=N_e}(X_j - x_j)^{\alpha}\phi^{'}(X_j - x_j) = \delta_{\alpha 0} \quad for\ \alpha = 0, ..., N_e \\ \sum_{i=1}^{i=N_e}(Y_j - Y_j)^{\alpha}\phi^{'}(Y_j - Y_j) = \delta_{\alpha 0} \quad for\ \alpha = 0, ..., N_e \\ \sum_{i=1}^{i=N_e}(Z_j - z_j)^{\alpha}\phi^{'}(Z_j - z_j) = \delta_{\alpha 0} \quad for\ \alpha = 0, ..., N_e \end{array} \tag{5.16}$$

In this work $N_e = 4$ is adopted to satisfy the constraints of Equation (5.10) and Equation (5.11). Taking the system of equations for the x-component as an example, this results in a moment-matrix taking the form of the well known Vandermonde matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ (X_j - x_1) & (X_j - x_2) & (X_j - x_3) & (X_j - x_4) \\ (X_j - x_1)^2 & (X_j - x_2)^2 & (X_j - x_3)^2 & (X_j - x_4)^2 \\ (X_j - x_1)^3 & (X_j - x_2)^3 & (X_j - x_3)^3 & (X_j - x_4)^3 \end{bmatrix} \begin{bmatrix} \phi^{'}(X_j - x_1) \\ \phi^{'}(X_j - x_2) \\ \phi^{'}(X_j - x_3) \\ \phi^{'}(X_j - x_4) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.17}$$

Considering the above system in the form of $Ax = B$, the weights $x = A^{-1}B$. Due to the form of B, the weights x are equal to the first column of A^{-1} which, as it's a Vendermonde matrix can be explicitly calculated. The weights can be calculated

explicitly as follows:

$$\begin{bmatrix} \phi^{'}(X_j - x_1) \\ \phi^{'}(X_j - x_2) \\ \phi^{'}(X_j - x_3) \\ \phi^{'}(X_j - x_4) \end{bmatrix} = -1 \begin{bmatrix} \frac{x_2^{'} x_3^{'} x_4^{'}}{(x_1^{'} - x_2^{'})(x_1^{'} - x_3^{'})(x_1^{'} - x_4^{'})} \\ \frac{x_1^{'} x_3^{'} x_4^{'}}{(x_2^{'} - x_1^{'})(x_2^{'} - x_3^{'})(x_2^{'} - x_4^{'})} \\ \frac{x_1^{'} x_2^{'} x_4^{'}}{(x_3^{'} - x_1^{'})(x_3^{'} - x_2^{'})(x_3^{'} - x_4^{'})} \\ \frac{x_1^{'} x_2^{'} x_3^{'}}{(x_4^{'} - x_1^{'})(x_4^{'} - x_2^{'})(x_4^{'} - x_3^{'})} \end{bmatrix} \tag{5.18}$$

where $x_1^{'} = (X_j - x_1)$, $x_2^{'} = (X_j - x_2)$, $x_3^{'} = (X_j - x_3)$ and $x_4^{'} = (X_j - x_4)$. x_2 and x_3 are the centroids of the cells that are closest to the Lagrangian node X_j and x_1 and x_4 are the centroids of the two neighbouring cells to x_2 and x_3 in the x-direction. As mentioned before a binary search is used to find x_2 and x_3, which is of order $O(log_2(number\ of\ x\ cells))$ if a structured non-uniform grid is used. A similar approach is then applied to calculate $\phi^{'}(Y_j - y)$ and $\phi^{'}(Z_j - z)$.

5.5 Benchmark Flow Problems

5.5.1 Introduction

In this section, two benchmark flow problems will be discussed that are used to test the efficacy of the LBFS-IBM. The flow over a cylinder in a square channel was chosen to test uniform IBM and shear flow over a sphere was chosen to test the non-uniform IBM. For all test cases the RMS of the residual was used to monitor convergence to steady-state and is defined as follows:

$$R_{RMS}\left(\mathbf{Q}^{Iteration,t}\right) = \sqrt{\frac{\sum_{i=1}^{No\ of\ Cells} \mathbf{R}\left(\mathbf{Q}_{i,t}\right)^2}{No\ of\ Cells}} \tag{5.19}$$

where t is the iteration index and $\mathbf{R}\left(\mathbf{Q}_j\right)$ is calculated from Equation (2.26). Equation (5.19) is normalised relative to the initial residual of the calculation giving the following convergence criterion:

$$\frac{R_{RMS}\left(\mathbf{Q}^{Iteration,t}\right)}{R_{RMS}\left(\mathbf{Q}^{Iteration,1}\right)} < 1 * 10^{-5} \tag{5.20}$$

In all flow problems, the ρu-momentum residual was chosen to monitor convergence as it was the slowest of both the mass and momentum residuals to converge.

As discussed previously an explicit forcing feedback scheme was adopted for the calculation of the force at each node in the Lagrangian object. As both flow problems involve fully fixed rigid objects, they use the same formulation for calculating the force. In this scheme the rigidity constraint is modelled by connecting each of the object nodes to a reference location with a spring of stiffness k. The reference location is simply the initial position of the object node $\mathbf{R}(t = 0)$. The force per area is then calculated as:

$$\mathbf{F}_a(\mathbf{R}(t), t) = -k[\mathbf{R}(t) - \mathbf{R}(0)] \tag{5.21}$$

As the object moves with the fluid, a restoring force is calculated and attempts to return the object to the reference location. The force at each node increases until an equilibrium is reached and a local no-slip condition is achieved. The stiffness k is a user defined parameter; if it is too small it results in too large a deformation and if it is too large it can result in stability issues.

5.5.2 Flow Over a Circular Cylinder in a Square Channel

5.5.2.1 Problem Set-Up

Flow over a circular cylinder in a square channel is a well established benchmark for investigating the performance of FSI CFD codes with complex geometries. Schäfer et al. [12] collated the predictions of ten different CFD codes for this flow problem and this work adopted the same geometry and flow conditions as these numerical studies. The fluid flows in a square channel and this was modelled by uniform Eulerian cells. A circular cylinder was immersed in the channel and this cylinder was modelled by Lagrangian nodes. The dimensions of the geometry are as described in Figure 5.3. As illustrated, the channel height and width $H = 0.41m$, the diameter of the cylinder $D = 0.1m$ and the characteristic velocity $\mathbf{V}_{RE} = 4\mathbf{V}(0, H/2, H/2)/9$. The top, bottom, front and back boundaries of the channel are set as no-slip boundary conditions. The right boundary is set to an outflow boundary by setting the pressure equal to atmospheric pressure. Finally

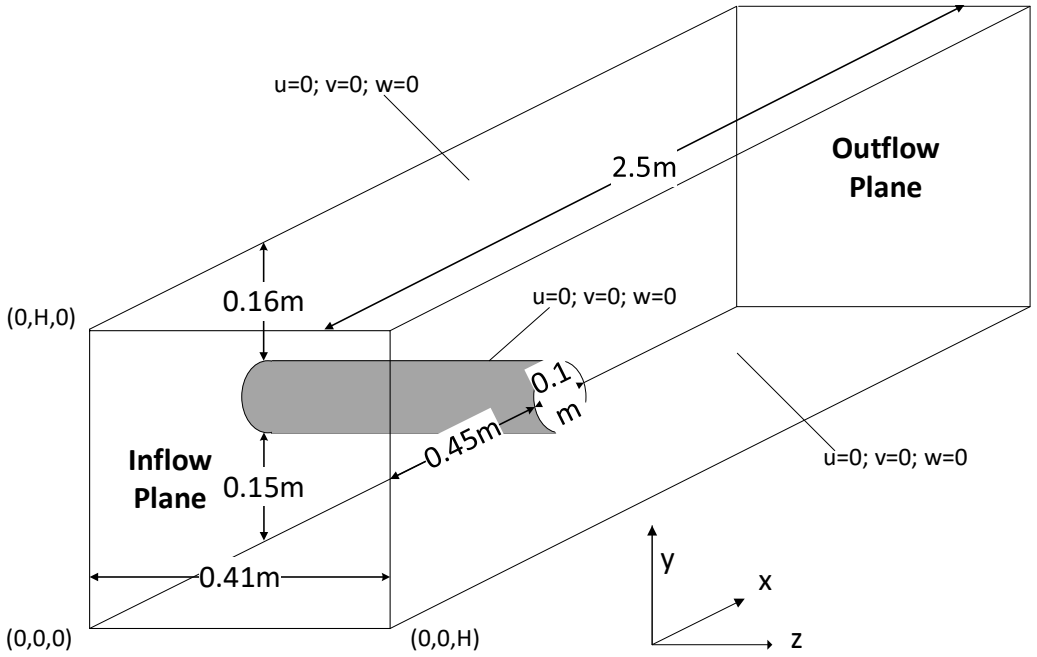


FIGURE 5.3: Problem set-up for flow over a cylinder in a square channel.

the left boundary is set equal to an inflow velocity of:

$$\mathbf{V} = \begin{bmatrix} 16u_m yz(H-y)(H-z)/H^4 \\ 0 \\ 0 \end{bmatrix} \tag{5.22}$$

where $u_m = 0.45\ m/s$. With a constant fluid density of 1.0 kg/m^3 and a kinematic viscosity $\nu = 10^{-3}\ m^2/s$ gives a Reynolds number $Re = 20$. The force in each cylinder node was calculated using Equation (5.21) and the area of each node was calculated by:

$$\Delta A(\mathbf{R}, t) = \pi D/N_{radial} * H/N_{axial} \tag{5.23}$$

where N_{radial} is the number of object nodes in the radial direction and N_{axial} is the number of object nodes in the axial direction. In this flow problem the influence of varying the stiffness k, the number of object nodes $N_{object\ nodes}$ and the number of Eulerian cells N_{cells} was investigated. The test cases that were run are shown in Table 5.1.

Test Case No.	Re	N_{cells}	$N_{object\ nodes}$	k
1	20	40000	900	1
2	20	87000	900	1
3	20	168000	900	1
4	20	400000	900	1
5	20	400000	900	0.01
6	20	400000	900	0.1
7	20	400000	900	1
8	20	400000	900	10
9	20	400000	400	100
10	20	400000	900	100
11	20	400000	1600	100
12	20	400000	2500	100
13	20	400000	100	1
14	20	400000	225	1
15	20	400000	400	1
16	20	400000	900	1
17	20	400000	1600	1
18	20	400000	2500	1

TABLE 5.1: Individual test case parameters varying with stiffness, N_{cells} and $N_{object\ nodes}$.

5.5.2.2 Results

The velocity contours and predicted streamlines for Test Case 18 are shown in Figure 5.4. As expected the streamlines are slightly asymmetric around the cylinder as the cylinder is positioned slightly off the mid-height of the channel. A recirculation region clearly develops in the lee of the cylinder and the no-slip boundaries are observed at the perimeter of the channel.

As mentioned before Schäfer et al. have collated the predictions for this flow problem from 10 different CFD codes. They used three parameters to measure the accuracy of the benchmark: the drag coefficient, the lift coefficient and the pressure differential between the front and back of the cylinder. The drag coefficient is calculated by:

$$C_d = \frac{F_d}{0.5\rho \mathbf{V}_{RE}^2 DH} \tag{5.24}$$

where F_d is the drag force given by:

$$F_d = - \sum_{j \in N_{object\ nodes}} \mathbf{F}_{a,j}(\mathbf{R}_j) \cdot \mathbf{i}\ \Delta A(\mathbf{R}_j) \tag{5.25}$$

The lift coefficient is given by:

$$C_l = \frac{F_l}{0.5\rho \mathbf{V}_{RE}^2 DH} \tag{5.26}$$

where F_l is the lift force given by:

$$F_l = - \sum_{j \in N_{object\ nodes}} \mathbf{F}_{a,j}(\mathbf{R}_j) \cdot \mathbf{j}\ \Delta A(\mathbf{R}_j) \tag{5.27}$$

and the pressure differential is defined by $\Delta P = P(x_{front}, y_{front}, z_{front}) - P(x_{back}, y_{back}, z_{back})$ where $(x_{front}, y_{front}, z_{front}) = (0.45, 0.20, 0.205)$ and $(x_{back}, y_{back}, z_{back}) = (0.55, 0.20, 0.205)$.

The results in Schäfer et al. are given in Table 5.2:

	C_d	C_l	ΔP
lower bound	6.05	0.008	0.1650
upper bound	6.25	0.01	0.1750

TABLE 5.2: Flow over a cylinder results from the literature [12].

The results in this work are given in Figures 5.5 to 5.7.

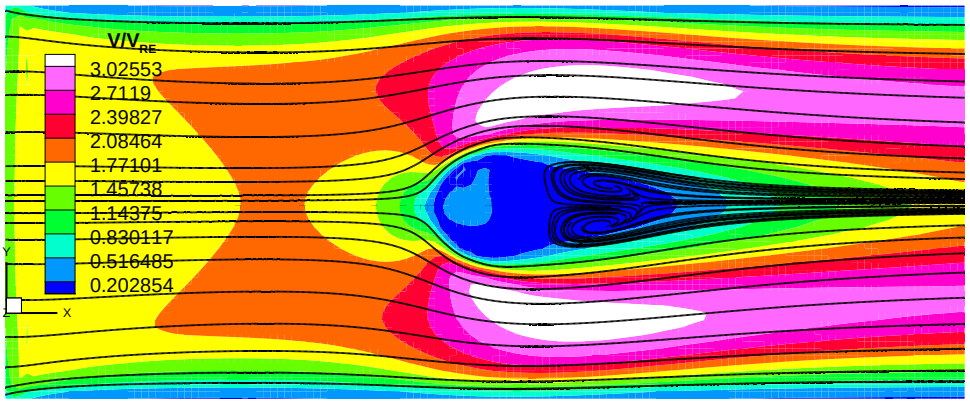
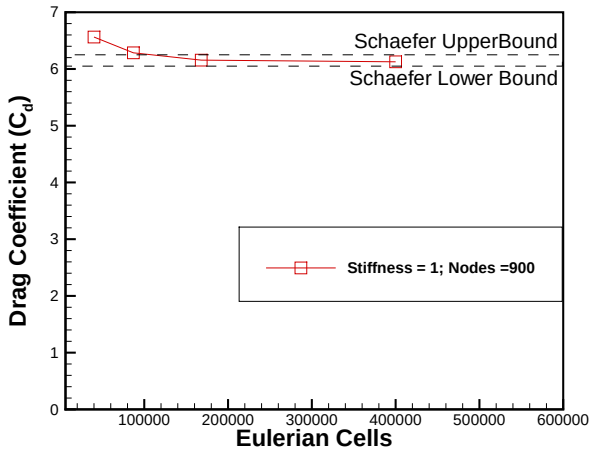
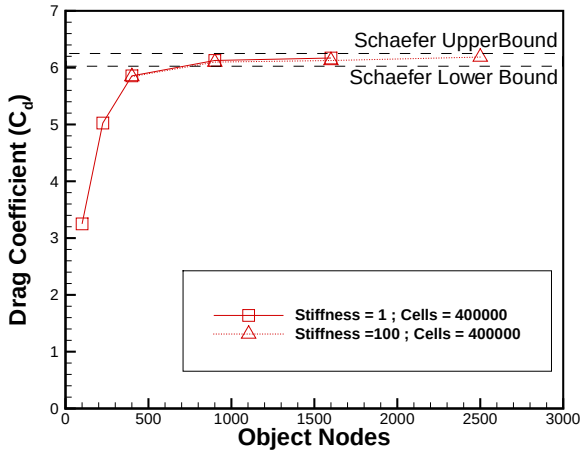


FIGURE 5.4: 3D flow over a circular cylinder in a square channel: streamlines and velocity contour plots for $Re = 20$, $N_{cells} = 400000$, $N_{object\ nodes} = 2500$ and $k = 1$.

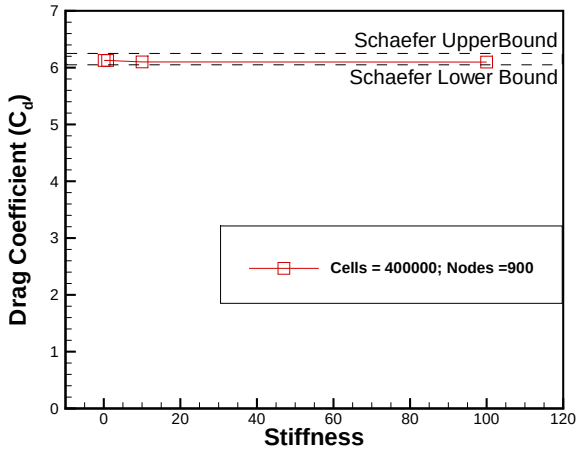
The results are well matched to the benchmarks provided by Schafer and Turek. In particular there is very good agreement for the drag coefficient and the pressure differential. It was also observed that increasing the mesh density and the object node density improves the accuracy of both these parameters. On the other hand



(a)

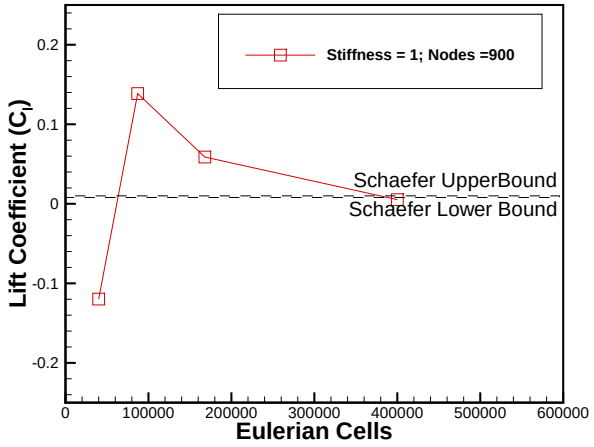


(b)

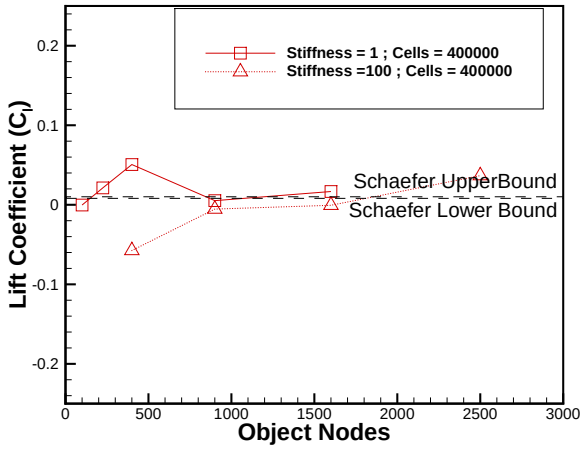


(c)

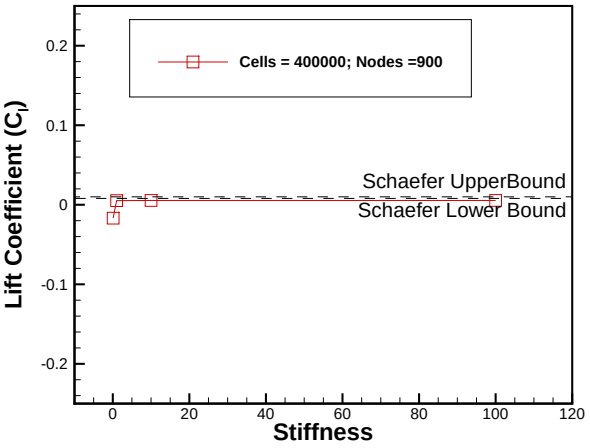
FIGURE 5.5: 3D flow over a circular cylinder in a square channel results for drag coefficient varying with (a) fluid cells (b) object nodes and (c) stiffness.



(a)

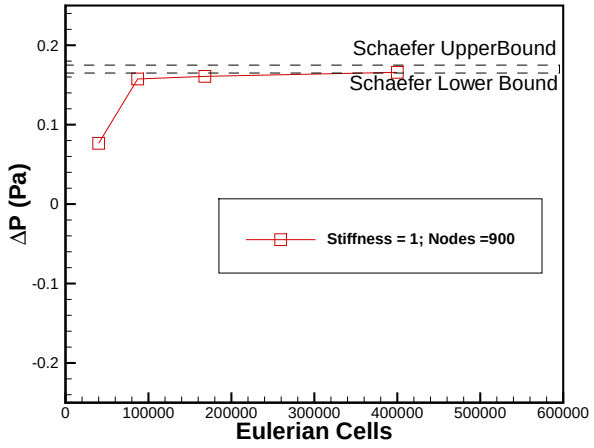


(b)

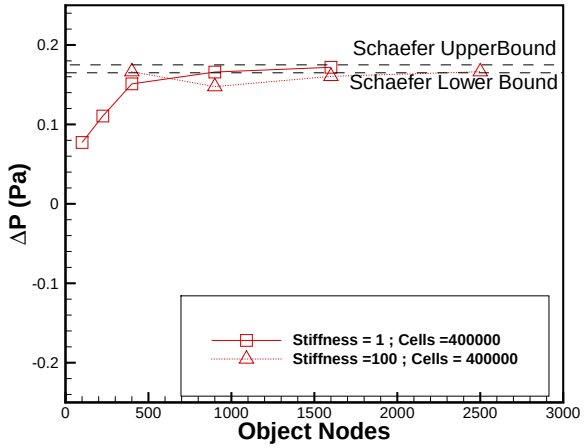


(c)

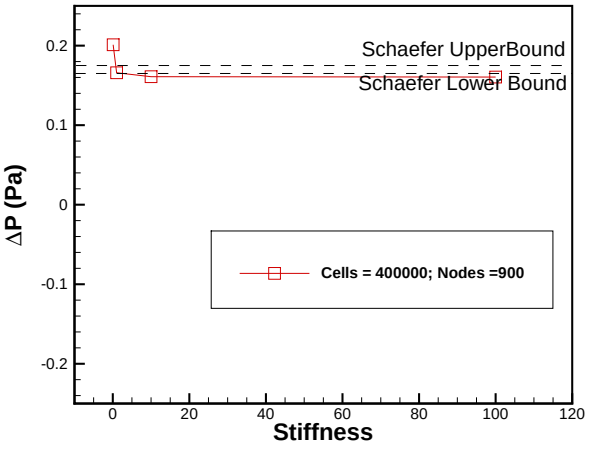
FIGURE 5.6: 3D flow over a circular cylinder in a square channel results for lift coefficient varying with (a) fluid cells (b) object nodes and (c) stiffness.



(a)



(b)



(c)

FIGURE 5.7: 3D flow over a circular cylinder in a square channel results for pressure differential ΔP varying with (a) fluid cells (b) object nodes and (c) stiffness.

the lift coefficient appears to be a more volatile parameter and sensitive to changes in the number of cells and object nodes. This is consistent with the work of Schafer and Turek; where there is massive inconsistency in the results for the lift coefficient. In some of the benchmarks in Schafer and Turek's study, there is negative values for the lift coefficient. Other benchmarks have increasing lift coefficient with mesh density and others have decreasing lift coefficient with mesh density. Finally there is little variation in the predictions of this work as the stiffness of the springs increases. This suggests that once a minimum level of rigidity is achieved, increasing the stiffness of the springs will not affect the results. To conclude, the results for this flow problem demonstrate that FSI is accurately modelled on uniform grids using the Peskin IBM.

5.5.3 Shear Flow Over a Sphere

5.5.3.1 Problem Set-Up

Linear shear flow over a solid sphere is a well established benchmark for investigating the performance of FSI CFD codes with complex geometries. There are many studies in the literature with the approach of Bagchi and Balachandar [206] is adopted in this work. The fluid flows in a cubic channel and this is modelled by a non-uniform mesh of Eulerian cells. A sphere is immersed in the channel and this sphere is modelled by Lagrangian nodes. The dimensions of the geometry are as described in Figure 5.8. As illustrated, the non-dimensional domain height,

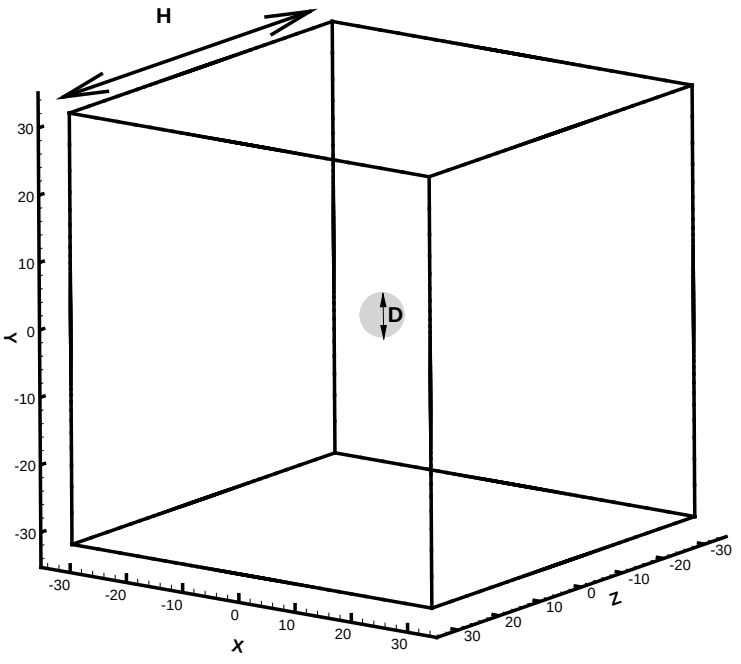


FIGURE 5.8: Problem set-up for shear flow over a sphere.

width and depth is $H = 64$ and the diameter of the cylinder is $D = 6.4$. The shear flow is assumed to be unbounded and in the absence of a sphere is set as:

$$\mathbf{V}(x, y, z) = \begin{bmatrix} Gy \\ 0 \\ 0 \end{bmatrix} \tag{5.28}$$

where G is the constant shear rate of the ambient flow along the y-axis. The front and back boundaries of the channel were set as zero-gradient boundaries. The bottom boundary was set as a no slip condition and the top boundary is a Dirichlet condition set equal to $\mathbf{V}(x, y, z)$ The right boundary was set to an outflow boundary by setting the pressure equal to atmospheric pressure. Finally the left boundary was set equal to an inflow velocity equal to $\mathbf{V}(x, y, z)$.

This flow problem requires two parameters to uniquely characterise the flow: the particle Reynolds number Re and also the shear Reynolds number Re_G. These are defined as:

$$\begin{aligned} Re &= \frac{\mathbf{V}_r D}{\nu} \\ Re_G &= \frac{GD^2}{\nu} \end{aligned} \tag{5.29}$$

where $\mathbf{V}_r$ is the relative velocity of the fluid to the sphere and is defined as $\mathbf{V}_r = \mathbf{V}(0, 0, 0) - \mathbf{V}_t$, where $\mathbf{V}_t$ is the instantaneous velocity of the sphere. Note that the two parameters are dependent and are related by:

$$\begin{aligned} Re_G &= s\ Re \\ s &= \frac{GD}{\mathbf{V}_r} \end{aligned} \tag{5.30}$$

The sphere mesh was created using GMSH and a tetrahedral mesh is generated containing 6559 nodes and 13112 triangles. The mesh is shown in Figure 5.9. The fluid computational domain is meshed using a non-uniform structured mesh. A plateau function was used to generate the distribution of the cell size in the x, y and z directions. The plateau function is described using the following formula:

$$f(x) = \left(\exp^{b(x-a)} + 1\right)\left(\exp^{b(-x-a)} + 1\right) \tag{5.31}$$

The plateau function contains two free parameters a and b. a indicates the size of the plateau at the centre of the grid where the cell size is approximately uniform. b indicates the steepness of the slopes of the curve outside the plateau. An example plot of the function is given in Figure 5.10. This function is then normalised with respect to both the number of nodes and the domain length to calculate the exact cell lengths in the x, y and z directions. In this flow problem 100 cells are used along each of the Cartesian axes and values $a = 32$ and $b = 0.2$. An

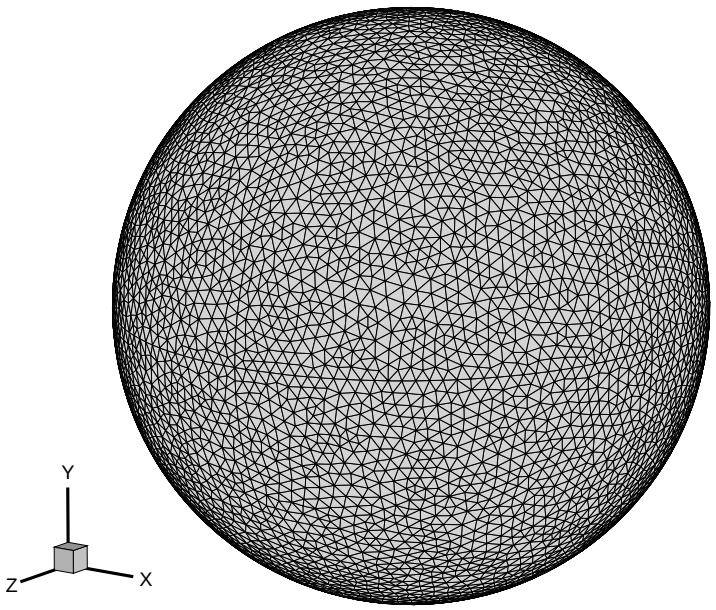


FIGURE 5.9: Tetrahedral mesh created using GMSH and containing 6559 nodes and 13112 triangles.

illustration of this mesh is provided in Figure 5.11. This results in cell heights of 0.2 in the vicinity of the sphere surface while using 10^6 cells in a mesh. An equivalent uniform mesh would require 32.76×10^6 cells. In this flow problem the range of parameters considered are $1 \leq Re \leq 200$ and $s = 0.4$ resulting in the following test cases being performed:

Test Case No.	Re	s
1	1	0.4
2	5	0.4
3	20	0.4
4	50	0.4
5	100	0.4
6	200	0.4

TABLE 5.3: Individual test case parameters varying with Re and s.

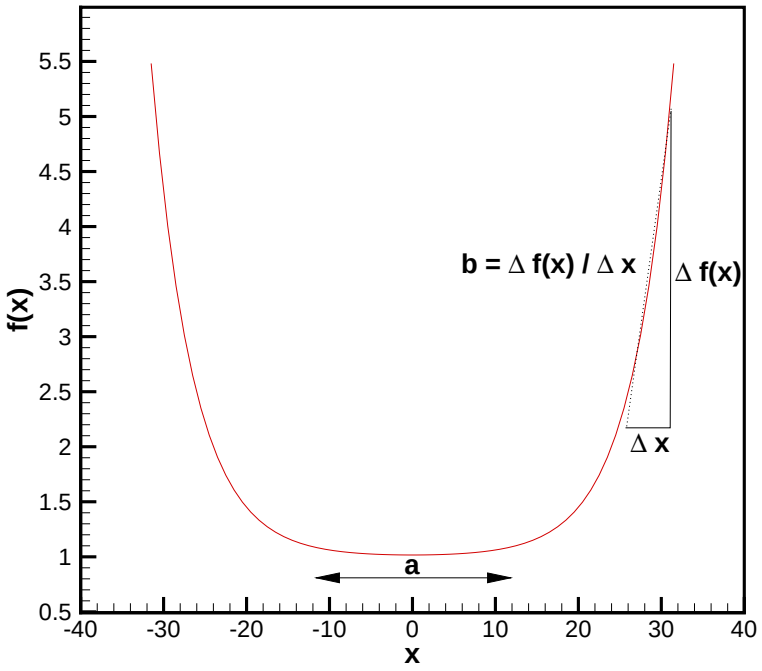


FIGURE 5.10: Plateau function where $a = 24$ and $b = 0.2$.

5.5.3.2 Results

The velocity contours and predicted streamlines for Test Case 5 are shown in Figure 5.12. Due to the asymmetric nature of the flow over the sphere, there is an asymmetric recirculation region observed in the lee of the sphere. Two parameters are used to measure the accuracy of the benchmark problem: the drag coefficient of the sphere and the lift coefficient of the sphere. These are calculated using Equations (5.24) to (5.27). The drag coefficient is compared to the standard drag law for uniform ambient flow provided by Schiller and Naumann [207] :

$$C_d = \frac{24}{Re}(1 + 0.15Re^{0.687}) \tag{5.32}$$

The lift coefficient is compared to the theoretical results of Saffman [208] and McLaughlin [209], the approximate expression by Mei [210], and the numerical simulations from Kurose and Komori [211] and Bagchi and Balachandar [206]. The latter two numerical simulations used body-fitted grids and finite difference and spectral methods respectively. The resulting drag coefficient is plotted in Figure 5.13 and the exponential curve presented in the work of Bagchi and Balachandar is recovered from the current work's predictions. Another similarity

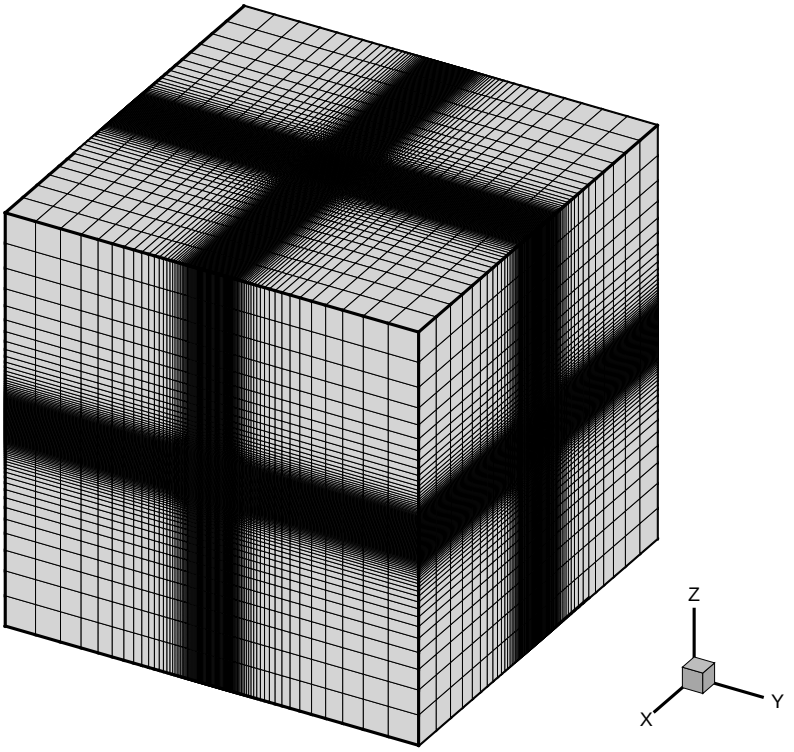


FIGURE 5.11: Mesh generated using a plateau function where $a = 24$ and $b = 0.2$

between the studies is that the drag coefficient is slightly larger than that predicted by the empirical formula as Re increases. The predicted lift coefficient is plotted in Figure 5.14 and again the exponential curve presented in the work of Bagchi and Balachandar is recovered from the current work's predictions. The predictions of lift coefficient in this work agrees well with the predictions of Bagchi and Balachandar and Kurose and Komori for $Re \geq 10$. Where as the values for $Re \leq 10$ are more closely aligned with that of Mei and Saffman. To conclude, there is good agreement between the present work and the results in the literature for this flow problem.

5.6 Summary

In this chapter an IBM for both uniform and non-uniform grids is introduced. Both methods were verified against known FSI benchmark flow problems and results matched very well with those in the literature. In particular it is shown that non-uniform structured grids can be used to reduce the number of cells required

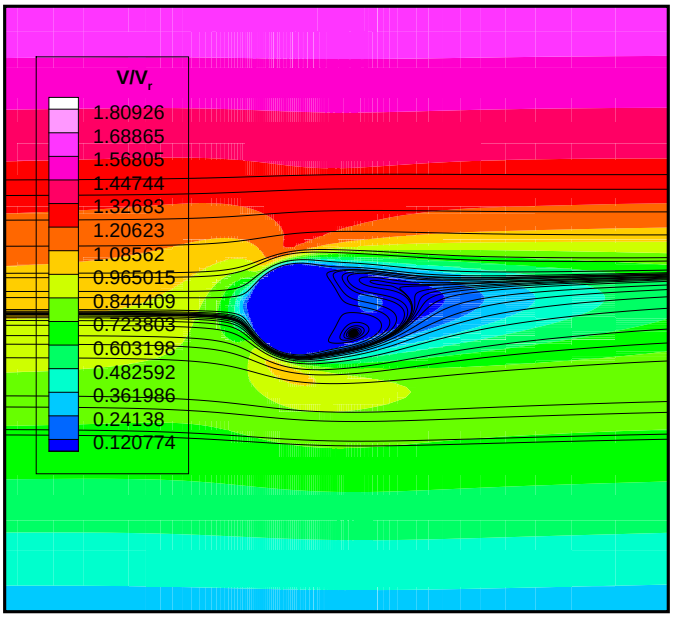


FIGURE 5.12: 3D sphere in shear flow: streamlines and velocity contour plots for $Re = 100$ and $s = 0.4$.

in the computational domain while offering accurate predictions of FSI problems. This is because non-uniform grids allow the use of grid refinement at a very cheap computational cost of a look up function to find the neighbouring fluid cells to the Lagrangian nodes.

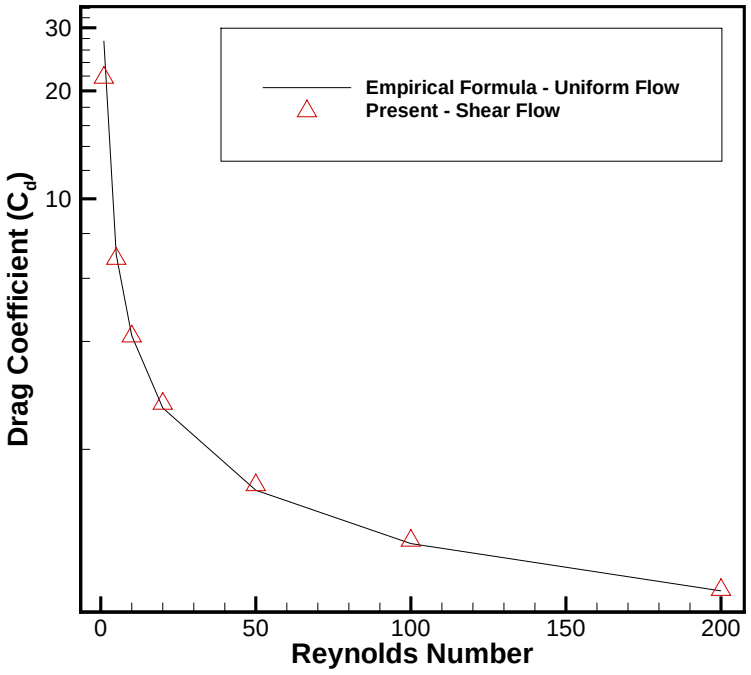


FIGURE 5.13: Drag coefficient for a sphere in shear flow for varying Reynolds numbers and $s = 0.4$.

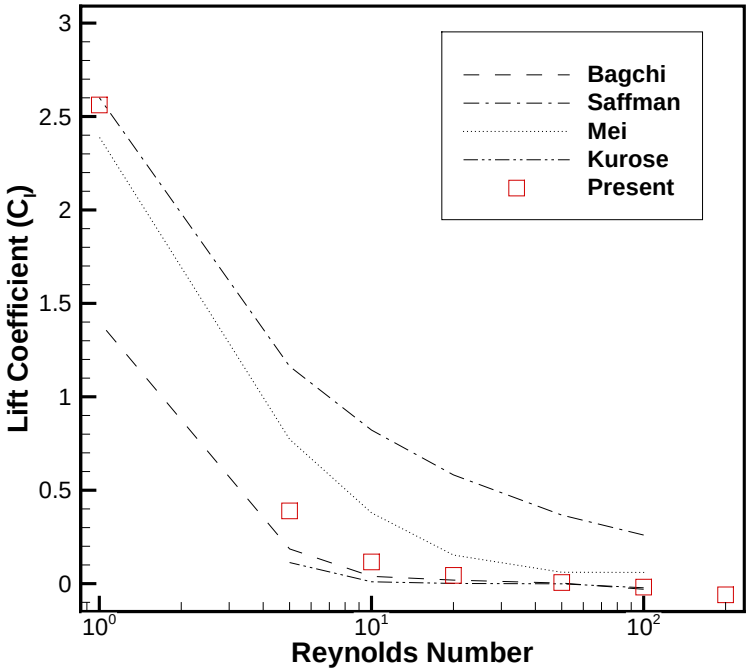


FIGURE 5.14: Lift coefficient for a sphere in shear flow for varying Reynolds numbers and $s = 0.4$.

Chapter 6

Red Blood Cell Structural Model

6.1 Introduction

In the ADE-SP RBC model the RBC is modelled as a surface of interconnected WLC springs. In this work the RBC surface is discretised into triangles (see Figure 6.4) and the edges on each of the triangles are modelled as a WLC spring. A WLC spring envisions a continuously flexible isotropic rod that can either display flexible or stiff behaviour depending on the length of the WLC. Each WLC spring connects two nodes on the mesh, where each node is considered a spring-particle. An example of a spring network containing triangles, spring-particles and WLC springs is shown in Figure 6.1.

In this chapter all aspects of the ADE-SP RBC model used in this work are introduced. This includes: an introduction of the Helmholtz free energies and the associated conservative forces used to capture the structural behaviour of the RBC, the model for capturing the membrane viscosity, the approach to calculating RBC geometry and the spatial discretisation of the RBC surface, model configuration and RBC parameter selection, methodology for calculating the interior viscosity of the RBC (*i.e.* the cytosol), approach to time integration of velocities of each spring-particle and finally the non-dimensionalisation approach to the ADE-SP RBC model is introduced. As mentioned previously the model adopted is from the work of Chen and Boyle [65]. A concise summary of the model is given here but for a detailed explanation of this model, please refer to Chen [1].

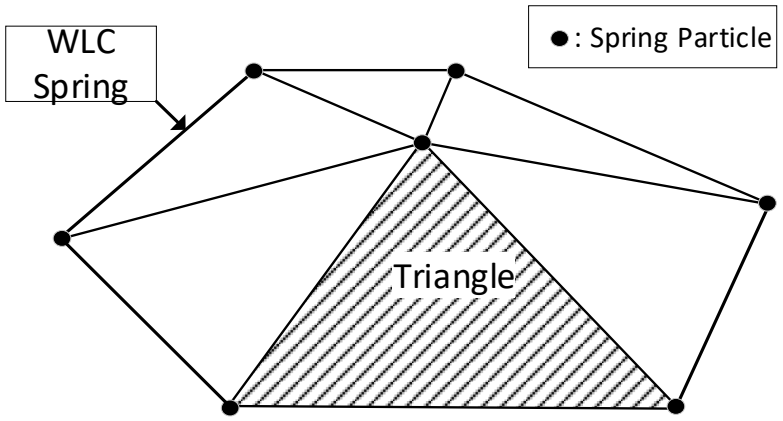


FIGURE 6.1: Illustration of spring network components including: triangles, spring-particles and WLC springs.

.

6.2 Helmholtz Free Energy and Internal Conservative Forces

6.2.1 Helmholtz Free Energy

The RBC consists of a surface membrane which surrounds an interior fluid. The fluid is known as cytosol and is an incompressible fluid which ensures that volume is conserved. The surface membrane consists of two main parts which confer structural strength to the RBC: the PM and the cytoskeleton. The PM provides structural resistance to surface compressibility and out-of-plane bending, while the cytoskeleton provides resistance to in-plane shearing. Each of these mechanical properties provides a corresponding contribution to the total Helmholtz energy of the RBC E_{RBC}, which is defined as:

$$E_{RBC} = E_V + E_A + E_B + E_S \tag{6.1}$$

where E_V is the energy due to the conservation of volume constraint, E_A is the energy due to the conservation of area constraint, E_B is the energy due to the out-of-plane bending resistance in the PM and E_S is the energy due to the in-plane shearing resistance of the cytoskeleton. E_{RBC} is the energy of the instantaneous shape of the RBC model.

6.2.2 Internal Forces

Stable equilibrium in a conservative system is defined as the equilibrium corresponding to minimum potential energy of the system [212]. The equilibrium shape of the RBC is the relaxed geometry *i.e.* the geometry of RBC with minimum energy. This is discussed in more detail in Section 6.4. Any RBC with a geometry other than the relaxed geometry of the RBC will have an excess energy that is used to bring the RBC back to its relaxed geometry. This conservative force is defined as:

$$\mathbf{F}(\mathbf{R}) = -\nabla E_{RBC}(\mathbf{R}) = -\nabla \left(E_V(\mathbf{R}) + E_A(\mathbf{R}) + E_B(\mathbf{R}) + E_S(\mathbf{R}) \right) \tag{6.2}$$

where $\mathbf{R}$ is the vector of particle coordinates in the RBC. Note that the force is in the direction that will cause the greatest reduction in free energy. Equation (6.2) also shows that the conservative force is dependent on the energy contribution from each of the four mechanical properties. To calculate the forces at each particle (*i.e.* mesh node) in the RBC, the contributions of the volume constraint, area constraint, bending energy and shearing energy have to be calculated. A summary of the equations required to calculate the force at each particle in the RBC and their discretised form is shown next.

6.2.3 Volume Constraint

The volume constraint energy is defined as follows:

$$E_V = \frac{K_V}{2V_{RBC,0}}(V_{RBC} - V_{RBC,0})^2 \tag{6.3}$$

where K_V is the volume constraint modulus, V_{RBC} is the instantaneous volume of the RBC and $V_{RBC,0}$ is the reference volume of the RBC. The conservative force due to the volume constraint is then defined as:

$$\mathbf{F}_{V,j} = -\nabla E_V = -\frac{\partial}{\partial \mathbf{R}_j}\left[\frac{K_V}{2V_{RBC,0}}(V_{RBC} - V_{RBC,0})^2\right] \tag{6.4}$$

where $\mathbf{F}_{V,j}$ is the conservative force due to the volume constraint, j is the particle index and $\mathbf{R}_j$ is the position of the j particle. The derivative of the instantaneous

volume is given by:

$$\frac{\partial V_{RBC}}{\partial \mathbf{R}_j} = \frac{1}{6}\sum_{k=1}^{N_k}\left(\frac{\partial \mathbf{m}_k}{\partial \mathbf{R}_j}\boldsymbol{\xi}_k + \frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_j}\mathbf{m}_k\right) \tag{6.5}$$

where k is the triangle index, N_k is the number of triangles in the RBC mesh, $\mathbf{m}_k$ is the centroid of triangle k and $\boldsymbol{\xi}_k$ is the normal to triangle k. Combining Equation (6.4) and Equation (6.5) gives the following:

$$\mathbf{F}_{V,j} = -\left[\frac{K_V(V_{RBC} - V_{RBC,0})}{6V_{RBC,0}}\right]\sum_{k=1}^{N_k}\left(\frac{\partial \mathbf{m}_k}{\partial \mathbf{R}_j}\boldsymbol{\xi}_k + \frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_j}\mathbf{m}_k\right) \tag{6.6}$$

For a discretised triangle such as Figure 6.2. The centroid, $\mathbf{m}_k$, can be calculated as:

$$\mathbf{m}_k = \frac{1}{3}\begin{bmatrix} x_1 + x_2 + x_3 \\ y_1 + y_2 + y_3 \\ z_1 + z_2 + z_3 \end{bmatrix} \tag{6.7}$$

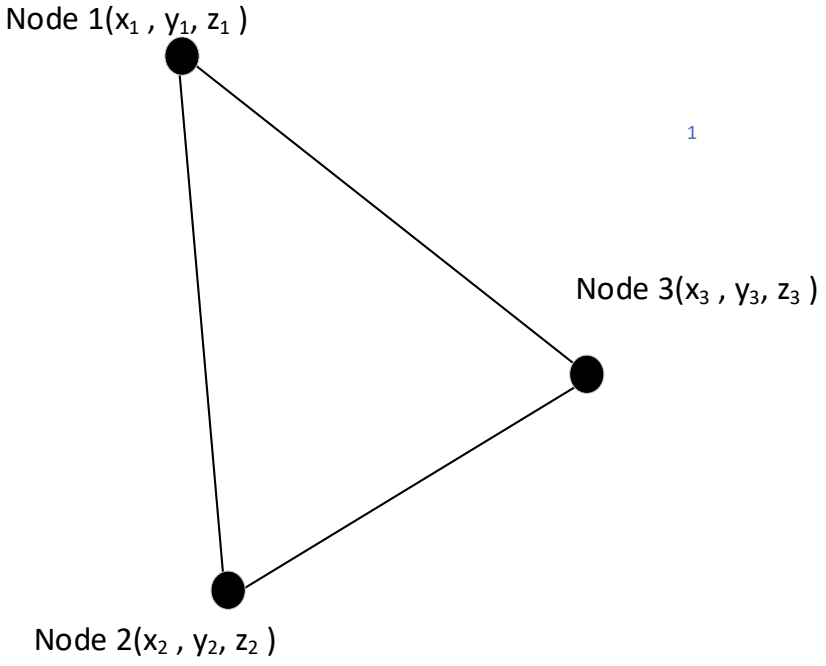


FIGURE 6.2: Sample triangle in discretised RBC mesh.

The normal to triangle k, $\boldsymbol{\xi}_k$, is given by:

$$\boldsymbol{\xi}_k = (\mathbf{R}_3 - \mathbf{R}_1) \times (\mathbf{R}_2 - \mathbf{R}_1) \tag{6.8}$$

The derivative of the centroid for Node 1 is given by:

$$\frac{\partial \mathbf{m}_k}{\partial \mathbf{R}_1} = \frac{\partial(\frac{1}{3}(\mathbf{R}_1 + \mathbf{R}_2 + \mathbf{R}_3))}{\partial \mathbf{R}_1} = \frac{1}{3} \tag{6.9}$$

similarly

$$\frac{\partial \mathbf{m}_k}{\partial \mathbf{R}_1} = \frac{\partial \mathbf{m}_k}{\partial \mathbf{R}_2} = \frac{\partial \mathbf{m}_k}{\partial \mathbf{R}_3} = \frac{1}{3} \tag{6.10}$$

The derivative of the normal for Node 1 is given by:

$$\frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_1} = \begin{bmatrix} 0 & z_3 - z_2 & y_2 - y_3 \\ z_2 - z_3 & 0 & x_3 - x_2 \\ y_3 - y_2 & x_2 - x_3 & 0 \end{bmatrix} \tag{6.11}$$

The derivative of the normal for Node 2 is given by:

$$\frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_2} = \begin{bmatrix} 0 & z_1 - z_3 & y_3 - y_1 \\ z_3 - z_1 & 0 & x_1 - x_3 \\ y_1 - y_3 & x_3 - x_1 & 0 \end{bmatrix} \tag{6.12}$$

The derivative of the normal for Node 3 is given by:

$$\frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_3} = \begin{bmatrix} 0 & z_2 - z_1 & y_1 - y_2 \\ z_1 - z_2 & 0 & x_2 - x_1 \\ y_2 - y_1 & x_1 - x_2 & 0 \end{bmatrix} \tag{6.13}$$

6.2.4 Area Constraint

The area constraint energy has a global and local contribution and is defined as follows:

$$E_A = \frac{K_A}{2A_{RBC,0}}(A_{RBC} - A_{RBC,0})^2 + \sum_{k=1}^{N_k} \left[\frac{K_{A,k}}{2A_{k,0}}(A_k - A_{k,0})^2 \right] \tag{6.14}$$

where K_A is the global area modulus, A_{RBC} is the instantaneous total area of the RBC, $A_{RBC,0}$ is the reference total area of the RBC, $K_{A,k}$ is the local area modulus, A_k is the instantaneous area of triangle k and $A_{k,0}$ is the reference area

of triangle k. The conservative force due to the area constraint is then defined as:

$$\mathbf{F}_{A,j} = -\nabla E_A = -\frac{\partial}{\partial \mathbf{R}_j}\left[\frac{K_A}{2A_{RBC,0}}(A_{RBC} - A_{RBC,0})^2 + \sum_{k=1}^{N_k}\left[\frac{K_{A,k}}{2A_{k,0}}(A_k - A_{k,0})^2\right]\right] \tag{6.15}$$

where $\mathbf{F}_{A,j}$ is the conservative force due to the area constraint. The derivative of the instantaneous total area is given by:

$$\frac{\partial A_{RBC}}{\partial \mathbf{R}_j} = \sum_{k=1}^{N_k}\left(\frac{\partial A_k}{\partial \mathbf{R}_j}\right) \tag{6.16}$$

where:

$$\frac{\partial A_k}{\partial \mathbf{R}_j} = \frac{1}{2}\left(\frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_j}\frac{\boldsymbol{\xi}_k}{|\boldsymbol{\xi}_k|}\right) \tag{6.17}$$

A_k is given by:

$$A_k = 0.5\ |\boldsymbol{\xi}_k| \tag{6.18}$$

Combining Equations (6.15) to (6.17) gives the following:

$$\mathbf{F}_{A,j} = -\frac{1}{2}\sum_{k=1}^{N_k}\left[\left(K_{A,k}\frac{(A_k - A_{k,0})}{A_{k,0}} + K_A\frac{(A_{RBC} - A_{RBC,0})}{A_{RBC,0}}\right)\frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{R}_j}\frac{\boldsymbol{\xi}_k}{|\boldsymbol{\xi}_k|}\right] \tag{6.19}$$

6.2.5 Bending Energy

The bending energy term has two contributions. The first is the Helfrich term which is due to local bending resistance. The second is the ADE term which is due to the contribution from the MAD. The bending energy is defined as:

$$E_B = \frac{K_B}{2}\sum_{j=1}^{N_j}\left[A_j \times (C_j - C_{j,0})^2\right] + \frac{K_{B,ADE}}{H_m^2 A_{RBC,0}}(\Delta A_m - \Delta A_{m,0}) \tag{6.20}$$

where K_B is the local bending modulus, N_j is the number of particles in the RBC mesh, A_j is the area of particle j, C_j is the instantaneous curvature of particle j, $C_{j,0}$ is the reference curvature of particle j, $K_{B,ADE}$ is the global bending modulus, H_m is the membrane thickness, ΔA_m is the instantaneous MAD of the RBC, and $\Delta A_{m,0}$ is the reference MAD of the RBC. The particle area A_j is defined as:

$$A_j = \frac{1}{3}\sum_{k=1}^{N_w} A_k \tag{6.21}$$

and the membrane curvature of the particle C_j is defined as:

$$C_j = \frac{\frac{1}{2}\sum_{s=1}^{N_w} \theta_s L_s}{A_j} \tag{6.22}$$

where N_w is the number of triangles/springs that share particle i, s is the spring index, θ_s is the angle between the two normals of the triangles which share spring s and L_s is the length of spring s. Refer to Figure 6.3 for an illustration of how A_s and C_s are calculated. Finally the instantaneous MAD is defined as:

$$\Delta A_m = H_m \sum_{j=1}^{N_j} C_j A_j \tag{6.23}$$

The conservative bending force is defined as:

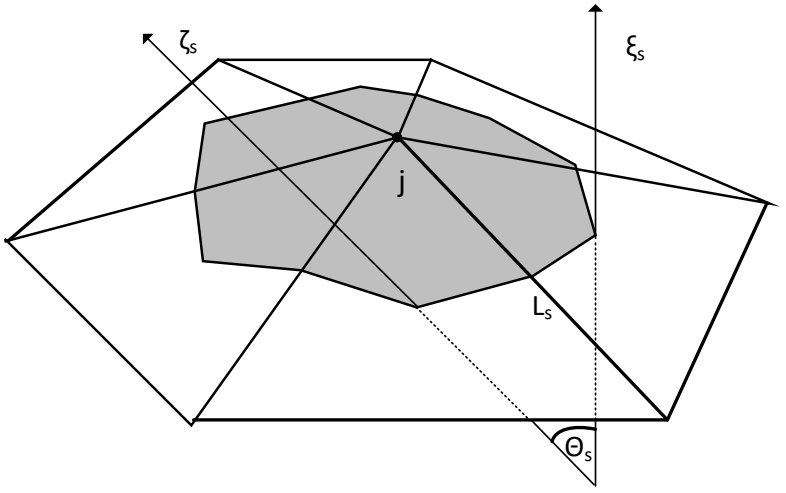


FIGURE 6.3: Illustration of particle area A_j and curvature C_j. A_j is equal to one third of the area of the triangles that share particle j; this is indicated by the grey area. C_j is dependent on the spring angle θ_s and spring length $\mathbf{L}_s$. θ_s is the angle formed by the normals ($\boldsymbol{\zeta}_s$ and $\boldsymbol{\xi}_s$) of the triangles which share spring s.

$$\begin{aligned}\mathbf{F}_{B,j} &= -\nabla E_B \\ &= -\frac{\partial}{\partial \mathbf{R}_j}\left[\frac{K_B}{2}\sum_{j=1}^{N_j}\left[A_j \times (C_j - C_{j,0})^2\right] + \frac{K_{B,ADE}}{H_m^2 A_{RBC,0}}(\Delta A_m - \Delta A_{m,0})\right]\end{aligned} \tag{6.24}$$

This can be re-written after much manipulation as:

$$\begin{aligned}\mathbf{F}_{B,j} = &-\sum_{j=1}^{N_j}\left[\left(K_B\frac{(C_j-C_{j,0})}{2}+K_{B,ADE}\frac{\Delta A_m-\Delta A_{m,0}}{H_m A_{RBC,0}}\right)\sum_{s=1}^{N_w}\frac{\partial\theta_s L_s}{\partial\mathbf{R}_j}\right]\\ &-\sum_{j=1}^{N_j}\left[K_B\frac{(C_j^2-C_{j,0}^2)}{6}\sum_{k=1}^{N_w}\frac{\partial A_k}{\partial\mathbf{R}_j}\right]\end{aligned} \tag{6.25}$$

where:

$$\frac{\partial\theta_s L_s}{\partial\mathbf{R}_j}=\frac{\partial\theta_s}{\partial\mathbf{R}_j}L_s+\frac{\partial L_s}{\partial\mathbf{R}_j}\theta_s \tag{6.26}$$

and:

$$\begin{aligned}\frac{\partial\theta_s}{\partial\mathbf{R}_j}=-\frac{1}{\sin\theta_s}\Bigg(&\frac{1}{|\boldsymbol{\xi}_s|}\left[\frac{\partial\boldsymbol{\xi}_s}{\partial\mathbf{R}_j}\left(\frac{\boldsymbol{\zeta}_s}{|\boldsymbol{\zeta}_s|}-\cos\theta_s\frac{\boldsymbol{\xi}_s}{|\boldsymbol{\xi}_s|}\right)\right]\\ &+\frac{1}{|\boldsymbol{\zeta}_s|}\left[\frac{\partial\boldsymbol{\zeta}_s}{\partial\mathbf{R}_j}\left(\frac{\boldsymbol{\xi}_s}{|\boldsymbol{\xi}_s|}-\cos\theta_s\frac{\boldsymbol{\zeta}_s}{|\boldsymbol{\zeta}_s|}\right)\right]\Bigg)\end{aligned} \tag{6.27}$$

and:

$$\frac{\partial L_s}{\partial\mathbf{R}_j}=\frac{\partial|\mathbf{L}_s|}{\partial\mathbf{R}_j}=\frac{\partial\mathbf{L}_s}{\partial\mathbf{R}_j}\frac{\mathbf{L}_s}{|\mathbf{L}_s|} \tag{6.28}$$

where $\mathbf{L}_s$ is the length of spring s in vector form and $\boldsymbol{\zeta}_s$ and $\boldsymbol{\xi}_s$ are the two normals to the triangles which share spring s. The discretised version of the derivative of the spring length for a spring between Node 1 and 2 in Figure 6.2 is given as:

$$\frac{\partial\mathbf{L}_{12}}{\partial\mathbf{R}_1}=\frac{(\mathbf{R}_2-\mathbf{R}_1)}{|\mathbf{R}_2-\mathbf{R}_1|} \tag{6.29}$$

6.2.6 Shear Energy

The shearing resistance in the spring-network is provided by the WLC springs employed in this RBC model. The shearing energy term is defined as:

$$E_S=\frac{k_B T_K}{2L_P}\sum_{s=1}^{N_s}\left[L_{c,s}\left(\frac{3\lambda_s^2-\lambda_s^3}{1-\lambda_s}-4c_1\lambda_s-c_2\right)\right] \tag{6.30}$$

where k_B is the Boltzmann constant, T_K is the absolute temperature in degrees Kelvin, L_P is the persistent length of the WLC springs, N_s is the total number of springs in the RBC mesh, $L_{c,s}$ is the contour length of the spring s and λ_s is the ratio of the instantaneous length of the spring to the contour length of the spring

i.e

$$\lambda_s = \frac{L_s}{L_{c,s}} \tag{6.31}$$

where L_s is the instantaneous length of the spring. The contour length is the maximum permissible length of a WLC and the persistence length is the length at which the WLC transitions from stiff to flexible behaviour.

The two constants in Equation (6.30) are required to set the equilibrium length of each spring. The standard WLC model tends to an equilibrium where $\lambda_s = \lambda_{s,0}$. The choice of value of equilibrium length is of much debate and is discussed in Section 6.4. To find the values of c_1 and c_2, the following constraints are set:

$$\begin{aligned} E_{WLC,s}(\lambda_s = \lambda_{s,0}) = 0 \\ F_{WLC,s}(\lambda_s = \lambda_{s,0}) = 0 \end{aligned} \tag{6.32}$$

where $E_{WLC,s}$ is the shear energy in spring s, $F_{WLC,s}$ is the force in spring s, $\lambda_{s,0}$ is the stretch ratio of the WLC at its reference length. Substituting into Equation (6.30) gives:

$$E_S(\lambda_{s,0}) = \frac{k_B T_K}{2L_P} \sum_{s=1}^{N_s} \left[L_{c,s} \left(\frac{3\lambda_{0,s}^2 - \lambda_{0,s}^3}{1 - \lambda_{0,s}} - 4c_1\lambda_{0,s} - c_2 \right) \right] = 0 \tag{6.33}$$

The general form of the conservative force due to shearing is:

$$\mathbf{F}_{S,j} = -\nabla E_S = -\frac{\partial}{\partial \mathbf{R}_j} \frac{k_B T_K}{2L_P} \sum_{s=1}^{N_s} \left[L_{c,s} \left(\frac{3\lambda_s^2 - \lambda_s^3}{1 - \lambda_s} - 4c_1\lambda_s - c_2 \right) \right] \tag{6.34}$$

which after some manipulation is equal to:

$$\mathbf{F}_{S,j} = -\frac{k_B T_K}{2L_P} \sum_{s=1}^{N_s} \left[\left(\frac{1}{4\left(1 - \lambda_s\right)^2} - \frac{1}{4} + \lambda_s - c_1 \right) \frac{\partial L_s}{\partial \mathbf{R}_j} \right] \tag{6.35}$$

Applying the constraint from Equation (6.32) to Equation (6.35) gives:

$$\mathbf{F}_{S,j}(\lambda_{s,0}) = -\frac{k_B T_K}{2L_P} \sum_{s=1}^{N_s} \left[\left(\frac{1}{4\left(1 - \lambda_{s,0}\right)^2} - \frac{1}{4} + \lambda_{s,0} - c_1 \right) \frac{\partial \lambda_{s,0}}{\partial \mathbf{R}_j} \right] = 0 \tag{6.36}$$

Solving Equation (6.36) and Equation (6.33) gives values of the shear constants as:

$$\begin{aligned} c_1 &= \frac{1}{4\left(1-\lambda_{s,0}\right)^2} - \frac{1}{4} + \lambda_{s,0} \\ c_2 &= \frac{3\lambda_{s,0}^2 - 2\lambda_{s,0}^3}{1-\lambda_{s,0}} - 4c_1\lambda_{0,s} \end{aligned} \tag{6.37}$$

The spring constant in Equation (6.35) for a spring s is given by:

$$\frac{k_B T_K}{L_P}(s) = \frac{4\lambda_{s,0} L_s K_S}{\sqrt{3}} \left(\frac{2(1-\lambda_{s,0})^3}{2(1-\lambda_{s,0})^3 + 1} \right) \tag{6.38}$$

where K_S is the shear modulus. Finally the derivative of the spring ratio is calculated from:

$$\frac{\partial \lambda_{s,0}}{\partial \mathbf{R}_j} = \frac{1}{L_{c,s}} \frac{\partial \mathbf{L}_s}{\partial \mathbf{R}_j} \tag{6.39}$$

6.3 Membrane viscosity

In the previous section the elastic behaviour of the RBC was described. This approach is adequate for static problems but in dynamic problems the PM exhibits viscoelastic behaviour. To account for the membrane viscosity, the approach of Ehi-Egharevba [213] is applied. The viscous force at each node is given by:

$$\mathbf{F}_{V,js} = -\mu_T \mathbf{V}_{js} - \mu_C (\mathbf{V}_{js} \cdot \frac{\mathbf{L}_s}{|\mathbf{L}_s|}) \frac{\mathbf{L}_s}{|\mathbf{L}_s|} \tag{6.40}$$

where $\mathbf{F}_{V,js}$ is the viscous force at particle j due to spring s, $\mathbf{V}_{js}$ is the relative velocity between particle j and the neighbouring particle on spring s, and μ_T and μ_C are coefficients related to the membrane viscosity μ_{PM} by the following relation:

$$\begin{aligned} \mu_{PM} &= \sqrt{3}\mu_T + \frac{\sqrt{3}\mu_C}{4} \\ \mu_C &= \frac{\mu_T}{3} \end{aligned} \tag{6.41}$$

6.4 Red Blood Cell Geometry and Spatial Discretisation

The relaxed shape of the RBC is the configuration of the RBC with minimum potential energy, *i.e.*

$$\nabla E_{RBC}(\mathbf{R}) = 0 \tag{6.42}$$

However, finding the initial relaxed discocyte is complicated by the fact that the reference values of the volume, area and bending energies come from the relaxed form of the PM, whereas the reference values of the shear energy comes from the relaxed form of the cytoskeleton. As detailed in Chen [1], the process to find the relaxed spatially discretised discocyte is summarised as follows:

- Create a sphere with reference area $A_{RBC,0}$ and volume V_{A0} by projecting the vertices of an icosahedron onto a sphere. V_{A0} is the volume of a sphere with area $A_{RBC,0}$.
- Find the relaxed ellipsoid shape of the cytoskeleton by reducing the volume of the sphere to $0.95V_{A0}$. This provides the reference lengths of the WLC springs, $L_{s,0}$, to find $\lambda_{s,0}$ using Equation (6.31).
- As shown by Peng et al. [214], the dimensionless parameter $c_0 = C_{j,0}R_{RBC,0}$ plays an important role in the final biconcave shape. c_0 is the reference curvature for each of the nodes. In this work $c_0 = 6$.
- Next find the relaxed discocyte shape of the RBC by reducing the volume of the sphere to $0.642V_{A0}$.

The initial sphere configuration is calculated by first deciding on a reference area $A_{RBC,0}$. The value used by Chen [1] of $A_{RBC,0} = 134\mu m^2$ is utilised in this work. The radius $R_{RBC,0}$ of the initial sphere is then given from:

$$R_0 = \sqrt{\frac{A_{RBC,0}}{4\pi}} \tag{6.43}$$

This gives $R_{RBC,0} = 3.265\mu m$. A regular icosahedron with 20 equilateral triangular faces is then created. To create a finer mesh, each equilateral triangle is then split into four equilateral triangles by using the midpoints of the edge of the triangle as vertices. Further subdivision can take place until the required mesh density is

acquired. Then each vertex on the icosahedron is radially projected onto a sphere of radius $R_{RBC,0}$. This gives a sphere with a mesh with a high level of homogeneity and a volume $V_{A0} = 145.85\mu m^3$.

The next step is to find the relaxed elliptical shape of the cytoskeleton by setting $V_{RBC,0} = 0.95V_{A0}$ as in Chen [1]. There is some debate over what the relaxed shape of the cytoskeleton is, with experimental evidence indicating a quasi-sphere shape [34] where-as numerical work would indicate an elliptical shape [215, 216]. All other reference values for area, bending and shear constraints are taken from the sphere configuration. From this relaxed elliptical shape, the reference lengths for the WLC springs $L_{s,0}$ for the next transformation are found.

Finally the relaxed discocyte shape is found by deflating the ellipsoid by setting $V_{RBC,0} = 0.642V_{A0}$ [215]. The reference area and reference lengths for the WLC springs are taken from the ellipsoid and the reference curvature is defined by setting $c_0 = 6$. This results in a discocyte shape as seen in Figure 6.4. In practice this mesh was found to have inconsistencies and did not behave symmetrically. These were removed by creating a biconcave mesh using the analytical formula proposed by Evans and Fung [217]:

$$z_{RBC} = \pm D_{RBC}\left(1 - \frac{4\left(x_{RBC}^2 - y_{RBC}^2\right)}{D_{RBC}^2}\right)^{0.5}\left(a_0 + a_1\frac{x_{RBC}^2 - y_{RBC}^2}{D_{RBC}^2} + a_2\frac{\left(x_{RBC}^2 - y_{RBC}\right)^2}{D_{RBC}^4}\right) \tag{6.44}$$

where x_{RBC}, y_{RBC} and z_{RBC} are the Cartesian coordinates of the mesh, D_{RBC} is the diameter of the RBC, and a_0, a_1 and a_2 are arbitrary constants set as $a_0 = 0.0518$, $a_1 = 2.026$ and $a_2 = -4.491$. This analytical mesh is then given the same reference values for volume, area, spring lengths and curvature as before. After setting these reference values, the analytical mesh moves to a new equilibrium which is the same as the equilbrium mesh deflated from the ellipsoid but without the spatial inconsistencies and symmetry errors. This is the initial shape used in all simulations in this work and can be seen in Figure 6.4.

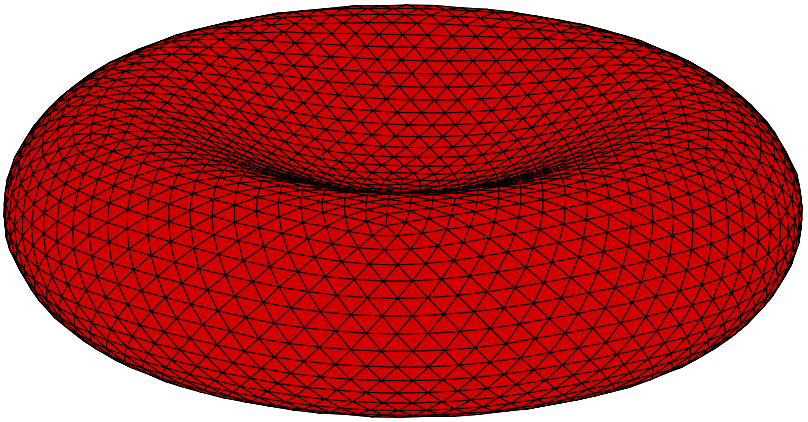


FIGURE 6.4: Discretised RBC discocyte with $V_{RBC,0} = 0.642V_{A0}$, $c_0 = 6$ and $N_j = 2562$.

.

6.5 Model Configuration

Many experiments have been performed to measure the various mechanical properties of a RBC. These measurements vary from study to study. The reasons for this include cell age, RBC nutrient levels, variations in experimental setup and experimental error [218]. This means that there is a range of observed values for RBC mechanical properties and that any properties adopted in this work must lie in this range. Where applicable, the values adopted by Chen were used [1]. These are presented in Table 6.1.

Note that physiological values of the Area and Volume modulus are of the order 10^9 N m^{-2}. These values are far in excess of those required to conserve area and volume. The adopted values of order 10^3 N m^{-2} is sufficient to conserve volume and area. As plasma is assumed to have the same properties as water, ρ_{plasma} is set equal to 1000 kg m^{-3}. $\mu_{cytosol}$ and μ_{PM} are adopted from the work of Swe Soe [219].

Property	Symbol	Values
Icosphere Radius	$R_{RBC,0}$	3.265 µm
RBC Area	$A_{RBC,0}$	134 µm^2
RBC Volume	$V_{RBC,0}$	93.641 µm^3
Volume Modulus	K_V	1000 N m^{-2}
Global Area Modulus	K_A	1000 N m^{-2}
Local Area Modulus	$K_{A,k}$	100 N m^{-2}
Local Bending Modulus	K_B	2.5×10^{-19} N m
Global Bending Modulus	$K_{B,ADE}$	2.5×10^{-19} N m
Shear Modulus	K_S	4×10^{-6} N m^{-1}
WLC Stretch Ratio	λ_s	2.6
Plasma Density	ρ_{plasma}	1000 kg m^{-3}
Cytosol Viscosity	$\mu_{cytosol}$	6 mPa s
Membrane Viscosity	μ_{PM}	0.7 µPa s m
Equilibrium Spring Length	$L_{s,0}$	Initial length on biconcave disc

TABLE 6.1: Properties of RBC adopted by Chen [1].

A key consideration here is that Chen, while using the ellipsoid for reference spring lengths to calculate the equilibrium shape, used the initial spring lengths of the biconcave disc when performing numerical simulations. This means that the RBC is stress free at equilibrium. The stress free shape of the RBC membrane is shown in the literature to have significant impacts on tank treading dynamics [214, 220, 221]. Using the biconcave discocyte as the stress free shape results in tank treading occurring at non-physiological values. Continuum RBC models have successfully used a nearly-spherical stress free shape when modelling tank treading [51, 222]. In comparison there is a lack of consistency in the literature with regards to spring-particle models. Pivkin [72] and Reasor [99] use a coarse graining procedure to specify an equilibrium length for every spring length. Fedosov adopts the same approach in his earlier work [223] but more recently uses a near spherical stress free shape as reference for spring equilibrium lengths [51]. Swe Soe [102] uses a biconcave stress free shape but does not perform tank treading. Finally Geekiyanage [216] uses a near spherical stress free shape but does not perform tank treading.

In this work the near spherical stress free shape was attempted for tank treading but resulted in unphysiological values of K_S for reference volumes $0.95V_{A0} <= V_{RBC,0} = 1.0V_{A0}$. Thereafter the approach of Pivkin was adopted and an average equilibrium length was set for each spring. In this approach K_S and $L_{s,0}$ are specified as input. K_S is then used to calculate the spring constant using Equation (6.38). $L_{s,0}$ specifies the equilibrium length of the spring and can be

used to set the springs in the initial discocyte mesh in compression or tension. This was then calibrated for the tank treading flow problem giving physiologically realistic values of $K_S = 6.1 \times 10^{-6}\mathrm{N\,m^{-1}}$ and $L_{s,0} = 2.85 \times 10^{-7}\mathrm{m}$. For comparison, Pivkin's coarse graining procedure would suggest a equilibrium length of $L_{s,0} = 2.28 \times 10^{-7}\mathrm{m}$. The RBC properties adopted in this work are given in Table 6.2.

Property	Symbol	Values
Icosphere Radius	$R_{RBC,0}$	3.265 µm
RBC Area	$A_{RBC,0}$	134 $\mu\mathrm{m}^2$
RBC Volume	$V_{RBC,0}$	93.641 $\mu\mathrm{m}^3$
Volume Modulus	K_V	1000 $\mathrm{N\,m^{-2}}$
Global Area Modulus	K_A	1000 $\mathrm{N\,m^{-2}}$
Local Area Modulus	$K_{A,k}$	100 $\mathrm{N\,m^{-2}}$
Local Bending Modulus	K_B	2.5×10^{-19} N m
Global Bending Modulus	$K_{B,ADE}$	2.5×10^{-19} N m
Shear Modulus	K_S	6.1×10^{-6} $\mathrm{N\,m^{-1}}$
WLC Stretch Ratio	λ_s	2.6
Plasma Density	ρ_{plasma}	1000 $\mathrm{kg\,m^{-3}}$
Cytosol Viscosity	$\mu_{cytosol}$	6 mPa s
Membrane Viscosity	μ_{PM}	0.7 µPa s m
Equilibrium Spring Length	$L_{s,0}$	2.85×10^{-7}m

TABLE 6.2: Properties of RBC adopted in this work.

6.6 Red Blood Cell Interior Viscosity

While the cytosol in the interior of the RBC membrane is considered to be an incompressible Newtonian fluid with the same density as the plasma,It has a different viscosity. Typically plasma has an exterior viscosity μ_{ext} of 1.2 mPa s while the cytosol has an interior viscosity μ_{int} of 6 mPa s. This gives a viscosity ratio $\lambda_\mu = \frac{\mu_{int}}{\mu_{ext}} \approx 5$. However in many experiments a fluid with a different viscosity to that of plasma is used, e.g. $\lambda_\mu \approx 0.27$ for a RBC suspended in a dextran solution. It has been shown that the behaviour of the RBC in shear flow is highly dependent on this ratio [224, 225]. As a result the blood flow solver must correctly assign viscosity values to the interior and exterior of the RBC during transient flow conditions. In this work a Heaviside function is used to assign viscosity values to fluid cells [226]. In this approach there is a gradual change in viscosity from the exterior to the interior to avoid discontinuities. For a given position in the fluid,

the viscosity is assigned as follows:

$$\mu\left(\mathbf{r}\right) = \mu_{ext} + (\mu_{int} - \mu_{ext}) H\left(\mathbf{r} - \mathbf{R}_{j,n}\right) \tag{6.45}$$

where $\mathbf{R}_{j,n}$ is the position of the nearest node in the RBC mesh to the fluid cell with position $\mathbf{r}$ and H is the Heaviside function given by:

$$H\left(\mathbf{r} - \mathbf{R}_{j,n}\right) = \begin{cases} 0.5\left[1 + \dfrac{\mathbf{r} - \mathbf{R}_{j,n}}{\Delta h} + \dfrac{1}{\pi}\sin\dfrac{\pi\left(\mathbf{r} - \mathbf{R}_{j,n}\right)}{\Delta h}\right] & \left|\mathbf{r} - \mathbf{R}_{j,n}\right| \leq \Delta h \\ 0 & \left|\mathbf{r} - \mathbf{R}_{j,n}\right| > \Delta h \text{ and external} \\ 1 & \text{otherwise} \end{cases} \tag{6.46}$$

where Δh is the edge length of the cells at the membrane interface. An example of viscosity distribution for a RBC using Equation (6.46) is shown in Figure 6.5.

6.7 Time Integration

The position of the nodes on the RBC membrane are simply updated by first interpolating velocities from the fluid onto the object nodes using Equation (5.2). These velocities are then integrated in time to advance the position of the membrane nodes. Due to the extremely small timescales involved, a forward Euler integration method is utilised. This is given as:

$$\mathbf{R}(t + \Delta t) = \mathbf{R}(t) + \mathbf{V}\left(\mathbf{R}, t\right)\Delta t \tag{6.47}$$

6.8 Non-Dimensionalisation

When validating the blood flow model, the results of numerical simulations are compared to real life experimental results. The latter are given in physical units whereas the numerical simulation uses non-dimensional units. Therefore a conversion factor is required for each input parameter. Letting a conversion factor be denoted by C and non-dimensional parameters be denoted by *, the conversion

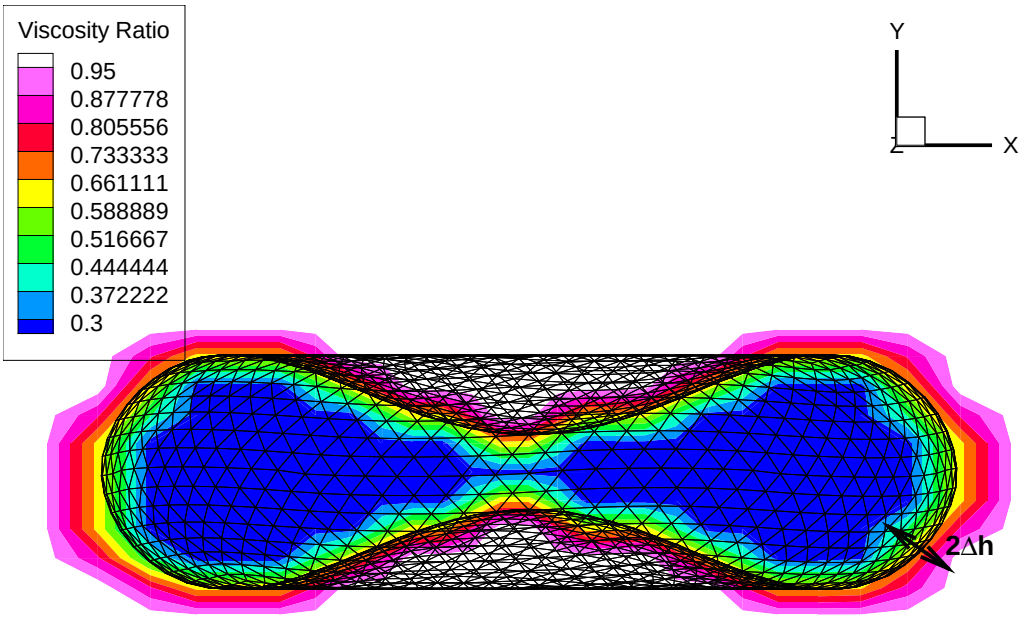


FIGURE 6.5: Viscosity ratio for immersed RBC in fluid where the internal viscosity $\mu_{int} = 0.27\mu_{ext}$.

factors for base SI units are:

$$C_{length} = \frac{R_{RBC,0}}{R_{RBC,0}*} = 1 \times 10^{-6} \tag{6.48}$$

$$C_{mass} = \frac{\rho_{plasma}}{\rho_{plasma}*} C_{length}^3 = 1000 \times 10^{-18} = 1 \times 10^{-15} \tag{6.49}$$

$$C_{time} = \frac{\Delta t}{\Delta t*} = 1 \times 10^{-5} \tag{6.50}$$

$$C_{force} = \frac{F}{F*} \frac{C_{mass} C_{length}}{C_{time}^2} = 1 \times 10^{-11} \tag{6.51}$$

Table 6.2 can now be converted to non-dimensional units and is shown in Table 6.3. The above approach can lead to excessive run times. Cimrak [227] suggests that the capillary number (Ca) is the most important dimensionless parameter when comparing different numerical experiments. This is because most isolated RBC flow problems have $Re << 1$ and the flow is laminar. Using the capillary number to

Property	Symbol	Physical Values	Non-Dimensional Values
Icosphere Radius	$R_{RBC,0}$	3.265 µm	3.265
RBC Area	$A_{RBC,0}$	134 µm^2	134
RBC Volume	$V_{RBC,0}$	93.641 µm^3	93.641
Volume Modulus	K_V	1000 N m^{-2}	100
Global Area Modulus	K_A	1000 N m^{-2}	100
Local Area Modulus	$K_{A,k}$	100 N m^{-2}	10
Local Bending Modulus	K_B	2.5×10^{-19} N m	0.025
Global Bending Modulus	$K_{B,ADE}$	2.5×10^{-19} N m	0.025
Shear Modulus	K_S	6.1×10^{-6} N m^{-1}	0.61
WLC Stretch Ratio	λ_s	2.6	2.6
Plasma Density	ρ_{plasma}	1000 kg m^{-3}	1
Cytosol Viscosity	$\mu_{cytosol}$	6 mPa s	60
Membrane Viscosity	μ_{PM}	0.7 µPa s m	4000

TABLE 6.3: Non-dimensional properties of RBC adopted in this work.

non-dimensionalise the RBC parameters allows the use of higher non-dimensional Re and reduces runtimes significantly. To follow this approach, the following equation must hold:

$$Ca = \frac{G\mu_{fluid}R_{RBC,0}}{K_S} = \frac{G * \mu_{fluid} * R_{RBC,0}*}{K_S*} \tag{6.52}$$

It will be indicated in each flow problem if the latter approach is taken.

6.9 Summary

This chapter has introduced the ADE-SP RBC model. In summary, the structural behaviour of the RBC is described by the internal conservative forces. These are the derivatives of the Helmholtz free energies for volume constraint, area constraint, bending resistance and shear resistance. The impacts of membrane viscosity on dynamic behaviour is captured with a viscous force. The reference volume, area and curvatures are taken from an icosphere whereas the reference spring lengths are directly specified. A Heaviside function is used to capture the variation in viscosity between cytosol and the external fluid. Finally the values of physical parameters used in the ADE-SP RBC model are given in dimensional and non-dimensional form.

Chapter 7

Red Blood Cell Validation

7.1 Introduction

In this chapter the blood flow model incorporating a LBFS, a non-uniform grid IBM and an ADE-SP RBC model is validated against experimental results for individual RBCs. The validation involved performing numerical simulations for a variety of benchmark experimental results. These experiments include:

- Optical Tweezers Test
- In-Plane Shear Flow - Wheel Configuration
- Out-of-Plane Shear Flow - Tumbling, Tank Treading and Swinging

During this work, all three experiments were first performed using the stress free RBC. It was only during the tank treading experiment that it was discovered that a pre-stressed RBC was required to recover physiologically realistic results. The optical tweezers and wheel configuration was then repeated with the pre-stressed configuration to confirm it's suitability for modelling blood flow.

7.1.1 Common Mesh Types

Each of the three benchmark flow problems had a common approach employed to meshing the suspending fluid. A RBC is immersed in a cubic fluid domain

with varying edge length. The fluid is meshed using a non-uniform structured mesh with a varying number of cells. A plateau function is used to generate the distribution of the cell size in the x, y and z directions. Please refer to Section 5.5.3 for details of plateau functions.

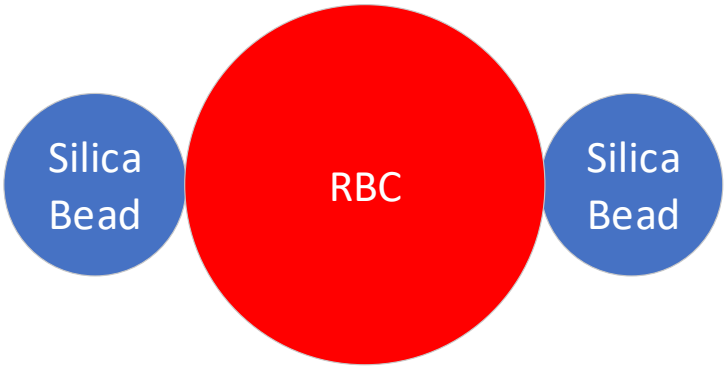
7.2 Optical Tweezers Test

7.2.1 Introduction

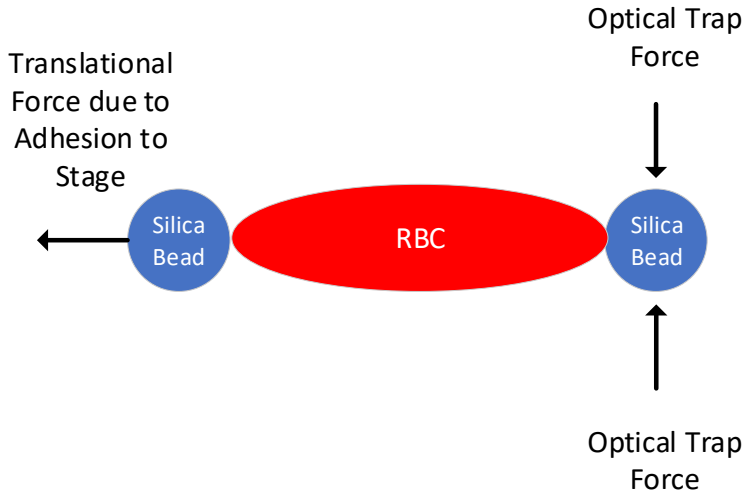
The optical tweezers experiment was performed to investigate the mechanical properties of RBCs in response to stretching by Mills et al. [6]. The RBCs in this experiment were procured from healthy adults using a lancet device and isolated using a centrifugation process. Two silica micro beads of 4.12 µm in diameter were then bound to diametrically opposite points on the RBC. This took place in a suspending medium of phosphate-buffered-saline (PBS) with PH 7.4 In this experiment the optical tweezers comprised a single-bead gradient optical trap. This involved the use of a laser to optically trap one of the silica beads. The second silica bead was then adhered to the glass slide on the stage. To apply a force the stage was then translated and a resistance force up to a maximum of 193 ± 20 pN was provided by the laser (see Figure 7.1 for an illustration of the experiment).

7.2.2 Problem Set-Up

The following approach was taken to numerically simulate this experiment. The computational domain used is cubic with edge length of 24 µm. The mesh was generated as described in Section 7.1.1 with 40 cells used along each of the Cartesian axes and plateau function values of $a = 12$ and $b = 0.2$. An illustration of this mesh is provided in Figure 7.2. This results in cell heights of 0.3 µm in the vicinity of the RBC-fluid boundary while using 64000 cells in the mesh. An equivalent uniform mesh would require 512000 cells. The initial RBC mesh has 2562 nodes and its initial shape is taken as a relaxed biconcave discocyte shape described in Section 6.4 and is positioned at the centre of the fluid domain as per Figure 7.2. Chen [1] showed that there was very little variation in results for the optical tweezers test with mesh density once a minimum of 289 nodes was



(a)



(b)

FIGURE 7.1: Optical tweezers experiment performed by Mills et al. [6] . (a) shows the initial conditions when two silica microbeads are attached to the RBC on diametrically opposed ends. (b) shows the mechanism for applying a stretching force to the RBC.

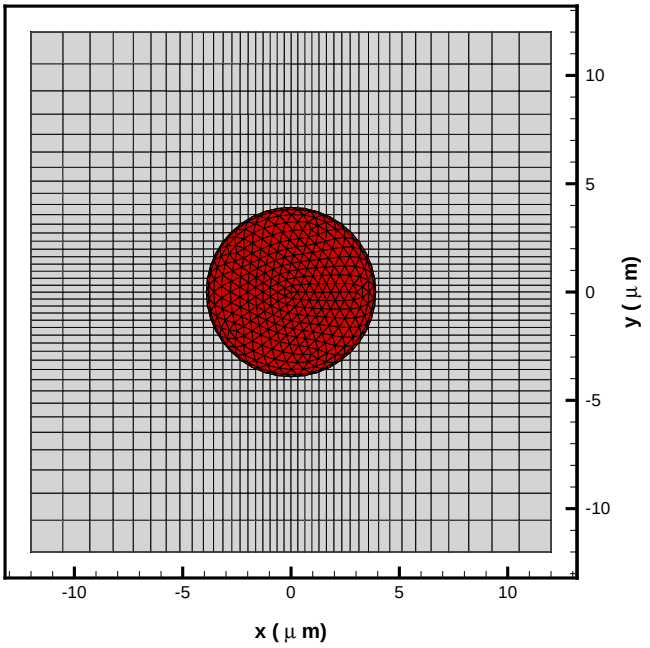


FIGURE 7.2: Initial mesh for optical tweezers test case. The fluid was meshed using a non-uniform grid where the node distribution is defined by a plateau function where $a = 24$ and $b = 0.2$. The RBC discocyte used is the relaxed shape of a 2562 node icosphere subject to Helmholtz free energy constraints.

used. PBS is considered to have similar mechanical properties to water and a viscosity ratio $\lambda_\mu \approx 5$ was used in this test case. The fluid has initial conditions of $\mathbf{V} = 0$ and $\rho_{fluid} = 1000\ \mathrm{kg\,m^{-3}}$. Neumann boundary conditions specifying zero gradient across the boundary were applied to all faces for both velocity and density. In Mills' experiment a range of stretching forces, $F_{Stretching}$, from 0 to 193 pN were applied to the RBC via the attached silica beads. The contact diameter D_c between the silica beads and the RBC was 2 µm [228]. $F_{Stretching}$ was then distributed over 4% of the nodes on the RBC mesh to reflect the contact diameter of 2 µm. This is 2% of the nodes with the largest values of x-coordinate and 2% of the nodes with the smallest values of x-coordinate. The absolute value of the force at each node $F_{Stretching,i} = \dfrac{F_{Stretching}}{0.02N_i}$ but orientated in opposite directions for maximum and minimum x-coordinate nodes (see Figure 7.3).

After $F_{Stretching}$ is applied, the simulation is run until equilibrium is reached. Static equilibrium is considered to be reached when the velocity of the maximum x-coordinate node, $i_{x,max}$, is less than $10^{-4}\ \mu\mathrm{m\,s^{-1}}$ and the velocity of the

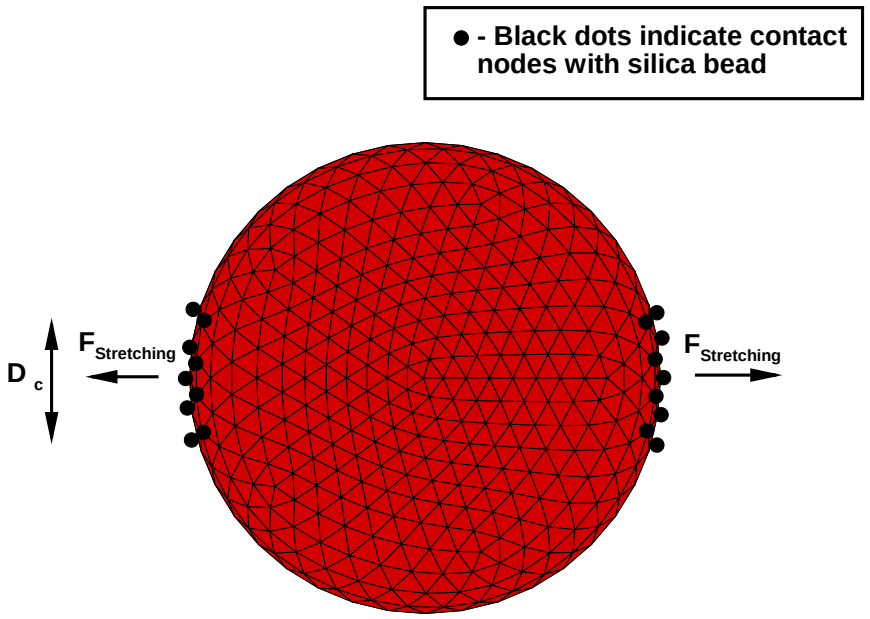


FIGURE 7.3: Initial conditions for simulating the application of the optical tweezer force to the RBC. The stretching force is applied to 2% of the nodes with the largest values of x-coordinate and 2% of the nodes with the smallest values of x-coordinate. This reflects the contact diameter of 2 µm of the silica bead with the RBC.

minimum x-coordinate node, $i_{x,min}$, is greater than -10^{-4} µm s^{-1}. Then the axial diameter $D_A = 2 \times max_{i \in N_i}(\sqrt{x_i^2 + y_i^2 + z_i^2})$ and the transverse diameter $D_T = 2 \times max_{i \in N_i}(\sqrt{y_i^2 + z_i^2})$ were calculated.

The simulation was performed twice, first using the RBC parameters for a stress free biconcave RBC from Table 6.1. This was done to show that the present work could recreate the results of Chen when the RBC was immersed in a suspending fluid. The simulation was then performed a second time with the parameters for a pre-stressed biconcave RBC from Table 6.2.

7.2.3 Results

The optical tweezers experiment was simulated in the range $F_{Stretching} = 0 - 200$ pN with a simulation performed at every multiple of 10 pN for a total of 20 simulations. The D_A and D_T at static equilibrium are plotted against $F_{Stretching}$ in Figure 7.5 and there is excellent agreement between the numerical and experimental results for the stress free biconcave RBC. In Figure 7.6, the final experimental and steady-state numerical results are plotted for a variety of forces. Again there

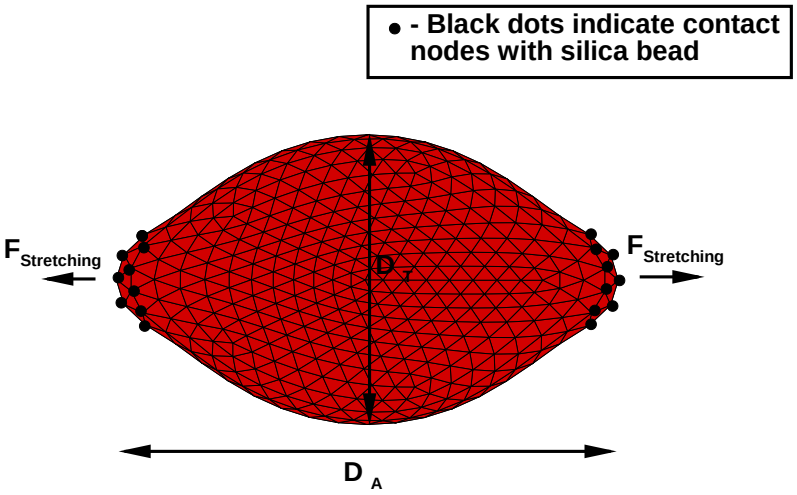


FIGURE 7.4: Static equilibrium of the RBC after the optical tweezers force is applied. The axial diameter D_A and transverse diameter D_T are indicated.

is excellent agreement between both sets of results. It is apparent that as the force increases that D_A increases and D_T decreases. The results for a pre-stressed biconcave RBC are also plotted in Figure 7.5. These results are not as accurate as the stress free biconcave RBC for the axial diameter but they are well within the error bars of the results of Mills. It is also noticeable that the pre-stressed biconcave RBC has better agreement with the experimental transverse diameter measurements.

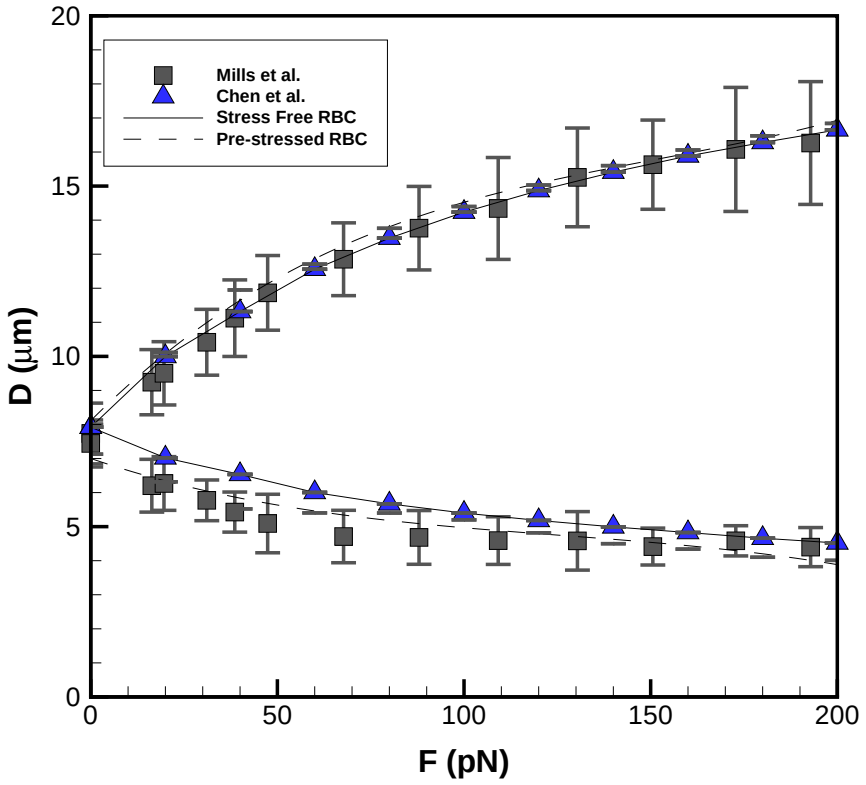


FIGURE 7.5: Plot of $F_{Stretching}$ with D_A (upper data) and D_T (lower data) at equilibrium for present work and Mills experimental results [6].

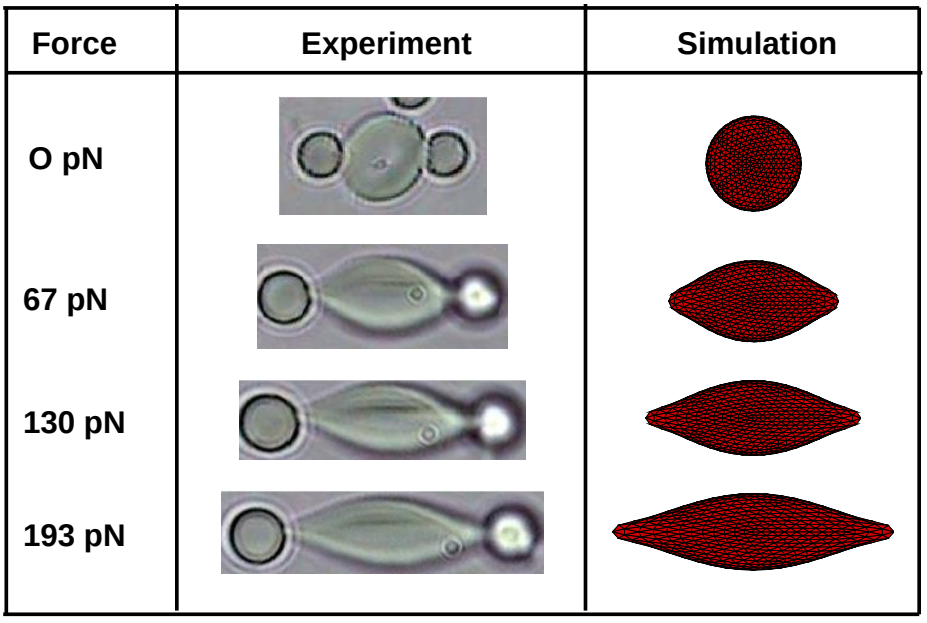


FIGURE 7.6: Comparison of present work with Mills experimental results [6] for static equilibrium shapes in optical tweezers experiment.

7.3 Deformation in "Wheel" Configuration

7.3.1 Introduction

Yao et al. [7] investigated the efficacy of Low Viscosity Ektacytometry (LVE) by measuring the deformation of RBCs in "wheel" configuration in a shear flow of low viscosity. The "wheel" configuration is when both major diameters of the biconcave RBC are in the same plane as the shear flow applied (see Figures 7.7 to 7.8). The measurements from LVE were compared to those produced by a flow chamber and imaging system by using a deformation index which is defined as:

$$DI = \frac{D_{max} - D_{min}}{D_{max} + D_{min}} \times 100 \tag{7.1}$$

where D_{max} and D_{min} are the lengths of the major and minor axes of the elliptical RBC. The imaging system involved taking 20 exposures of digital photographs of a RBC suspension in shear flow and measuring the DI directly from the photo of the RBCs in the "wheel" configuration. In comparison LVE involves measuring the DI by passing a laser through a RBC suspension in shear flow and measuring the elliptical diffractive pattern on a phototube. Experiments were performed for shear rates in the range $0\ \text{s}^{-1} < G < 120\ \text{s}^{-1}$. The suspending medium used was PBS with a viscosity ratio $\lambda_\mu \approx 8.48$. For both approaches a linear relationship between DI and G was discovered. In this validation problem these linear relationships are used to measure the accuracy of the ADE-SP RBC model.

7.3.2 Problem Set-Up

The following approach was taken to numerically simulate this experiment. A RBC was immersed in a cubic fluid domain with edge length of 24 µm. The fluid was meshed using a non-uniform structured mesh with 8000 cells. In this flow problem 20 cells were used along each of the Cartesian axes and plateau function values of $a = 8$ and $b = 0.2$. An illustration of this mesh is provided in Figure 7.8. This results in cell heights of 1.0 µm in the vicinity of the RBC-fluid boundary while using 8000 cells in the mesh. An equivalent uniform mesh would require 13824 cells. The initial RBC mesh has 2562 nodes and its initial shape was taken as a biconcave RBC described in Section 6.4 and was positioned at the centre of the fluid domain as per Figure 7.2. PBS is considered to have similar mechanical

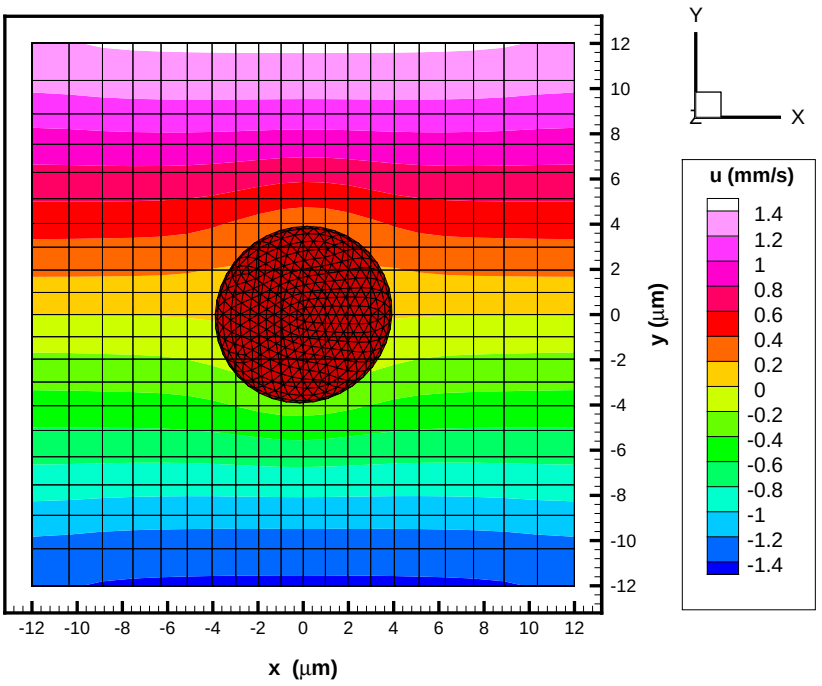


FIGURE 7.7: RBC in initial wheel configuration subject to shear flow $G = 120\ \mathrm{s}^{-1}$. Initially v and w are equal to 0.

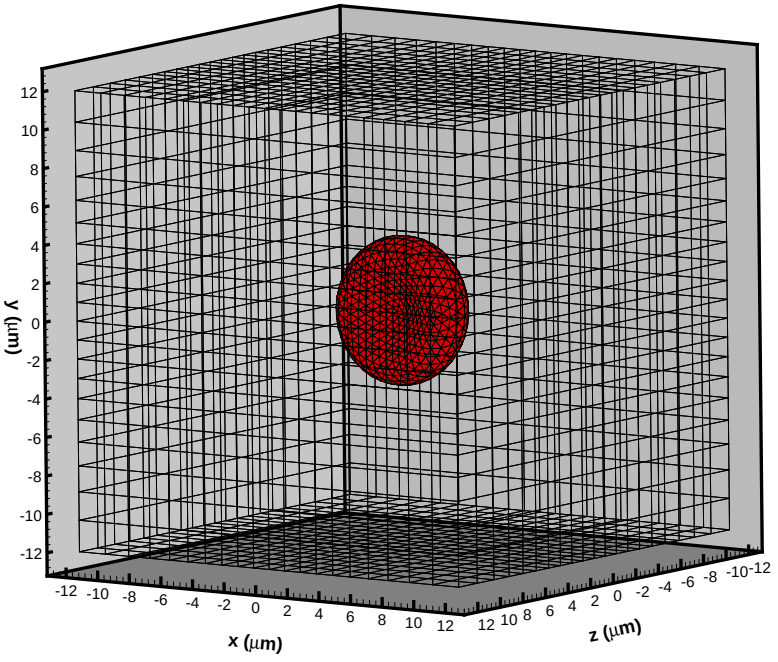


FIGURE 7.8: 3D view of RBC in initial wheel configuration.

properties to water and a viscosity ratio $\lambda_\mu \approx 8.48$ is used in this test case. The fluid had initial conditions of $\mathbf{V} = 0$ and $\rho_{fluid} = 1000\ \mathrm{kg\,m^{-3}}$. Dirichlet boundary conditions specifying a velocity equivalent to the shear rate G are applied to all faces. Neumann boundary conditions specifying zero gradient across the boundary are applied to all faces for density. In Yao's experiment, the RBCs are subjected to shear rates in the range $0\ \mathrm{s^{-1}} < G < 120\ \mathrm{s^{-1}}$. This resulted in the following linear relationships between DI and G:

$$DI_{FlowChamber} = 0.0805G - 0.3602 \tag{7.2}$$

$$DI_{LVE} = 0.0609G + 0.5837 \tag{7.3}$$

In this work, simulations were run in the range $0\ \mathrm{s^{-1}} < G < 120\ \mathrm{s^{-1}}$, with a simulation performed at every multiple of $20\mathrm{s^{-1}}$. The RBC had the material properties described in Table 6.1, however a sensitivity analysis was performed on the shear modulus K_S to investigate it's impact on DI. Simulations were performed in the range of $2\mu\mathrm{N\,m^{-1}} < K_S < 6.1\ \mu\mathrm{N\,m^{-1}}$. Simulations were also performed with the parameters for a pre-stressed biconcave RBC from Table 6.2.

7.3.3 Results

The initial simulation was run for 3 periods of revolution and it was found that 1.25 periods of revolution were sufficient to find a constant DI. The remaining simulations were run for 1.25 periods of revolution of the RBC. For each simulation the DI was calculated by extracting the x and y coordinates of the RBC mesh and finding the minimum sized rectangle that bounds these coordinates (see Figure 7.9). The minimum rectangle was calculated using a script which fits a convex hull to the xy coordinates and then calculates a bounding box for each vertex of the convex hull. The height and width of the rectangle of the bounding box with minimum area are then plugged into Equation (7.1) to calculate the DI. The resulting DI is compared to the linear relationships of Yao in Equations (7.2) to (7.3). The results are shown in Figure 7.10. As can be seen the simulated results show an approximate linear relationship between the DI and G for all values of K_S. This is in agreement with the experimental work done by Yao et al. [7]. However the results with $K_S = 4\mu\mathrm{N\,m^{-1}}$ with an initial stress free shape show a DI which is slightly lower than that of the experimental work. As seen from the sensitivity analysis, a value of $K_S = 3 \times 10^{-6}\ \mathrm{N\,m^{-1}}$ for a stress free shape

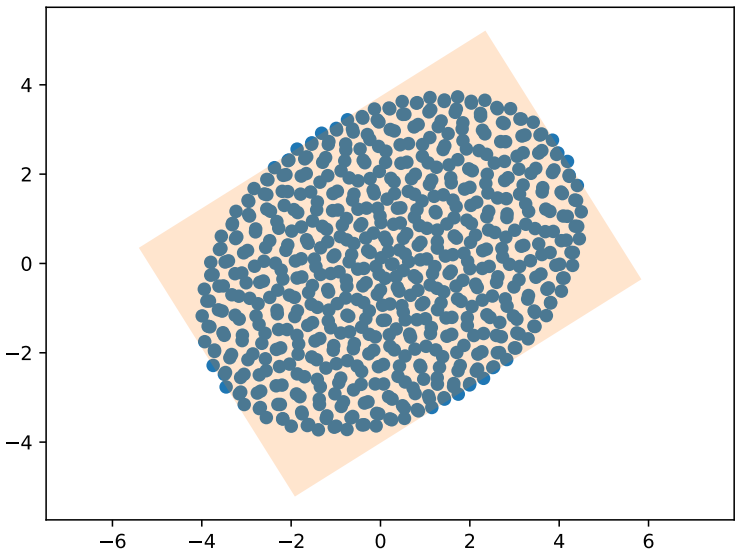


FIGURE 7.9: Rectangle of minimum area which bounds the x and y coordinates of the RBC mesh projected onto xy plane. The deformation index is then calculated from the height and width of the rectangle.

shows better agreement with the experimental work of Yao et al. [7]. When a pre-stressed initial shape was used with a value of $K_S = 6.1\mu\text{N m}^{-1}$, the deformation observed was much lower than the experimental results. They were however significantly more accurate than the stress free results with $K_S = 6.1\mu\text{N m}^{-1}$.

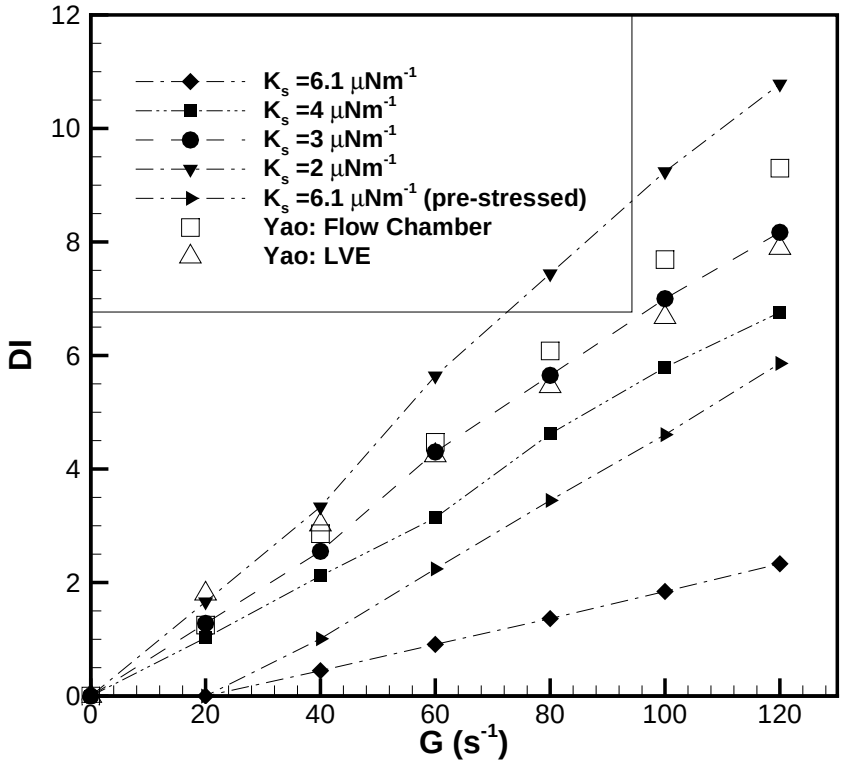


FIGURE 7.10: *DI* of a RBC in wheel configuration for varying shear rates and shear moduli. Results are shown from the present work and compared with the experimental measurements of Yao et al. [7].

7.4 Tumbling, Tank Treading and Swinging

7.4.1 Introduction

The dynamic behaviour of RBCs in shear flow, where the two major axes of the RBC are out of plane with the shear flow applied, has been the subject of many experimental studies over several decades. The observations in these studies have shown three primary dynamical modes: tumbling (TB) at low shear rates, tank treading (TT) at high shear rates and an intermediate regime when transitioning from TB to TT occurs [229, 230]. TB is where the pitch angle of the RBC, see Figure 7.11, undergoes continuous full 360 degree revolutions. TT is where the pitch angle remains approximately constant but the surface of the RBC undergoes a continuous revolution around the centroid of the RBC. The transition from TB to TT is due to the existence of "shape memory" in the RBC; this is the existence of a minimum energy when the RBC is in static equilibrium [231]. The transition to TT occurs when the shear stress applied by a fluid exceeds the maximum elastic energy storage of the RBC [232]. This transition point is a function of the shear rate G, viscosity ratio λ_μ of the RBC cytosol to the suspending fluid, and the elastic properties of the RBC. In experiments to date, isolated RBCs have been subjected to shear flow whilst suspended in various solutions of dextran. The solutions of dextran have varying viscosities and images were recorded of the dynamic behaviour of the RBCs. Abkarian et al. [10] used dextran solutions with viscosities of 22, 31 and 47 mPa s. Fischer [233] used dextran solutions with viscosities ranging from 12.9 to 102.9 mPa s. Abkarian et al. established a relationship between the oscillation period of the RBC and the shear rate. They also identified the critical shear stresses for the transition point from TB to TT. Fischer has also observed these characteristics in his studies. Numerical studies have been performed by Pivkin [8] and Reasor [9] which compare favourably to the experimental work of Abkarian et al.. Numerical simulations of a RBC in shear flow were performed in this work and compared to the experimental work of Abkarian et al. and the numerical work of Pivkin and Reasor.

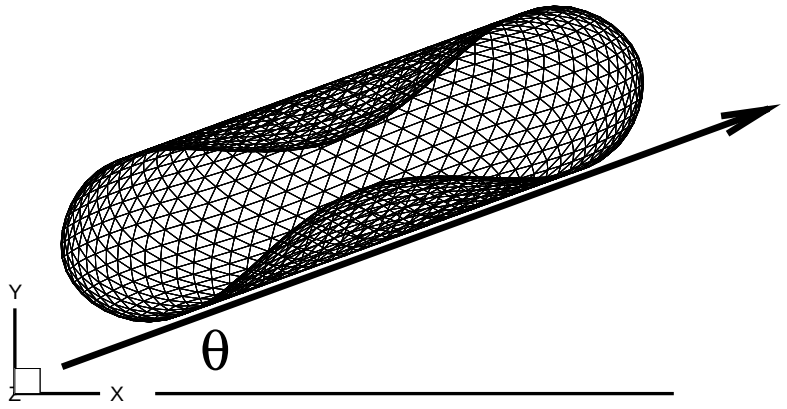


FIGURE 7.11: RBC pitch angle with xz Cartesian plane.

7.4.2 Problem Set-Up

The following approach was taken to numerically simulate this experiment. A RBC was immersed in a cubic fluid domain. The domain size was reduced compared to previous flow problems due to the increased computational cost of this problem. Various domain sizes were trialled and an edge length of 12 µm was found to be the smallest edge length that did not adversely impact on the simulations performed. The fluid is meshed using a non-uniform structured mesh with 27000 cells. In this flow problem 30 cells are used along each of the Cartesian axes and plateau function values of $a = 10$ and $b = 0.2$. An illustration of this mesh is provided in Figure 7.12. This results in cell heights of 0.25 µm in the vicinity of the RBC-fluid boundary while using 27000 cells in the mesh. An equivalent uniform mesh would require 110592 cells. The initial RBC mesh has 2562 nodes and its initial shape is taken as the pre-stressed biconcave RBC described in Section 6.4 and is positioned at the centre of the fluid as per Figure 7.12. A constant Mach number of 0.173 is used in each simulation to minimise run times. The shear Reynolds number Re_G varies between 0.39 and 0.025. Dextran is considered to have similar mechanical properties to water and the fluid has initial conditions of $\mathbf{V} = 0$ and $\rho_{plasma} = 1000\ \mathrm{kg\,m^{-3}}$. In this work, the approach of the experimental work of Abkarian et al. and the numerical work of Pivkin and Reasor is adopted where a suspending fluid with a viscosity of 22 mPa s is used. This gives a viscosity ratio of $\lambda_\mu = 0.27$. Dirichlet boundary conditions specifying a velocity equivalent to

the shear rate G are applied to all faces for all velocity components. Neumann boundary conditions specifying zero gradient across the boundary are applied to all faces for both velocity and density. In this work, simulations were run in the range 0 $s^{-1} < G < 7.5$ s^{-1}. The RBC has the material properties described in Table 6.2.

7.4.3 Results

The first benchmark parameter identified was the period of rotation of the RBC under varying shear rates. It was also identified if the RBC was TB or TT at each shear rate. The results are shown in Figure 7.13. As can be seen there is very good agreement between the present predictions and the predictions of Pivkin [8] and Reasor [9]. The present work correctly predicts that lowering the shear rate results in larger periods of oscillation of the RBC. The present work also predicts a transition between TB and TT in the range of 1 $s^{-1} < G < 1.4$ s^{-1}. Reasor predicts a range of 1 $s^{-1} < G < 1.5$ s^{-1} and Pivkin predicts a range of 1.15 $s^{-1} < G < 1.35$ s^{-1}.

Snapshots of a RBC tumbling ($G = 0.5$ s^{-1}) and a RBC tank treading ($G = 1.6$ s^{-1}) are shown in Figures 7.14 and 7.15. As is shown, the TB RBC fully rotates around the z Cartesian axis. In comparison, the TT RBC slightly oscillates around a fixed pitch angle from the xz Cartesian plane. This behaviour is referred to as swinging and the amplitude of this angle has been experimentally measured by Abkarian et al.for the case of $G = 1.8$ s^{-1}. The pitch angle is illustrated in Figure 7.11. In the present work, the pitch angle is calculated by applying a principal component analysis [234] to each xy coordinate on the RBC mesh. This gives a vector of the RBC from which the pitch angle can be calculated. The pitch angle of the case where $G = 1.8$ s^{-1} is shown in Figure 7.16. It can be seen that there is very good agreement between the present work and the experimental measurements of Abkarian et al. for the amplitude of the pitch angle. The initial period of the results should be discounted as the RBC starts from steady state and initial perturbations are removed. Abkarian et al. also discovered a logarithimic relationship between the amplitude of the swinging angle and the shear rate. The results from the present work are shown in Figure 7.17. The linear curve fitted to the results has a slope of -0.983 which is in excellent agreement with Abkarian et al.'s slope of -1. Finally intermittent behaviour was also observed in the present

work for $G = 1.2$ s^{-1} and is shown in Figure 7.18. As can be seen, the RBC alternates between TB and TT behaviour. Abkarian et al. observes intermittent behaviour at a much higher value of $G = 1.526$ s^{-1}. This was not observed in the current work. The numerical works of Reasor and Pivkin do not show results for intermittent behaviour of a RBC but claim the transition/intermittency zone is between $1 \text{ s}^{-1} < G < 1.5 \text{ s}^{-1}$ and $1.15 \text{ s}^{-1} < G < 1.35 \text{ s}^{-1}$ respectively.

In conclusion the present work shows that ADE-SP LBFS blood flow model can model the complex RBC dynamics of TB, TT and the intermittent region. The results produced by the present work are in great agreement with the experimental and numerical results in the literature. One interesting feature is that the results produced were unsensitive to changes in membrane viscosity. This would indicate that the membrane viscosity approach used by Ehi-Egharevba [213] is unsuitable when a RBC is immersed in a fluid and another approach should be used.

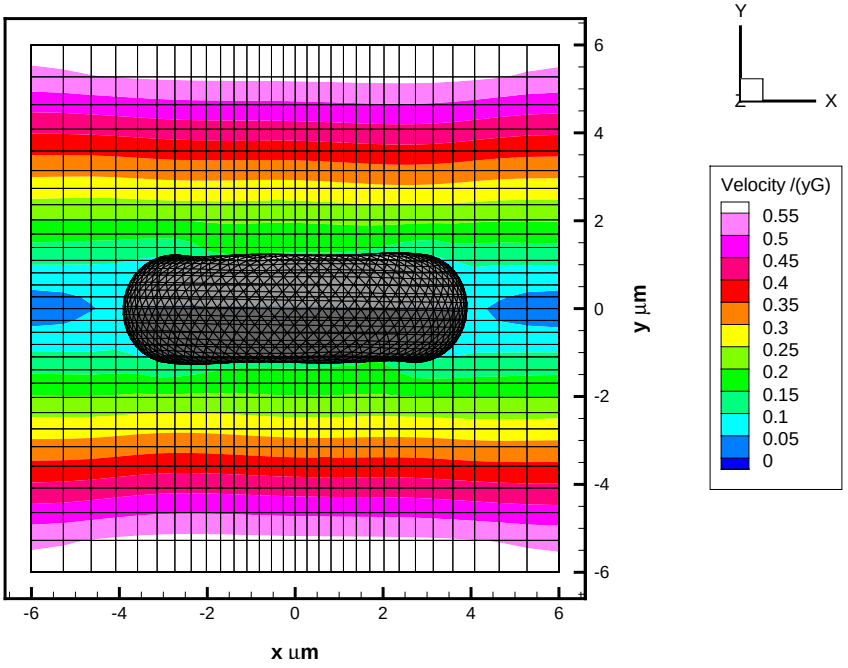


FIGURE 7.12: RBC in initial tank treading configuration subject to shear flow $G = 0.5\ \mathrm{s}^{-1}$. Initially v and w are equal to 0.

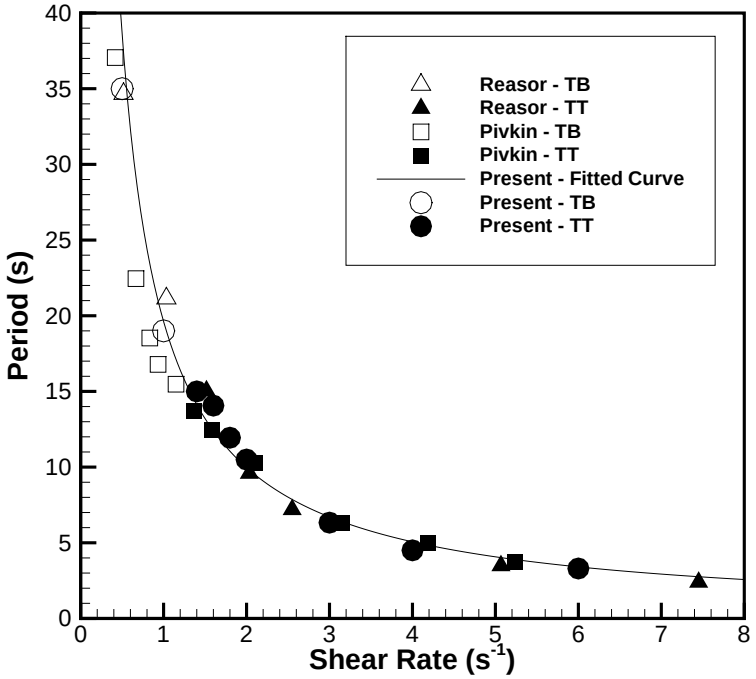


FIGURE 7.13: Comparison of shear rate and period of rotation of a RBC for $\lambda_\mu = 0.27$. Results are shown for the current work and the numerical work of Pivkin [8] and Reasor [9]. The mode of the rotation of the RBC is also denoted.

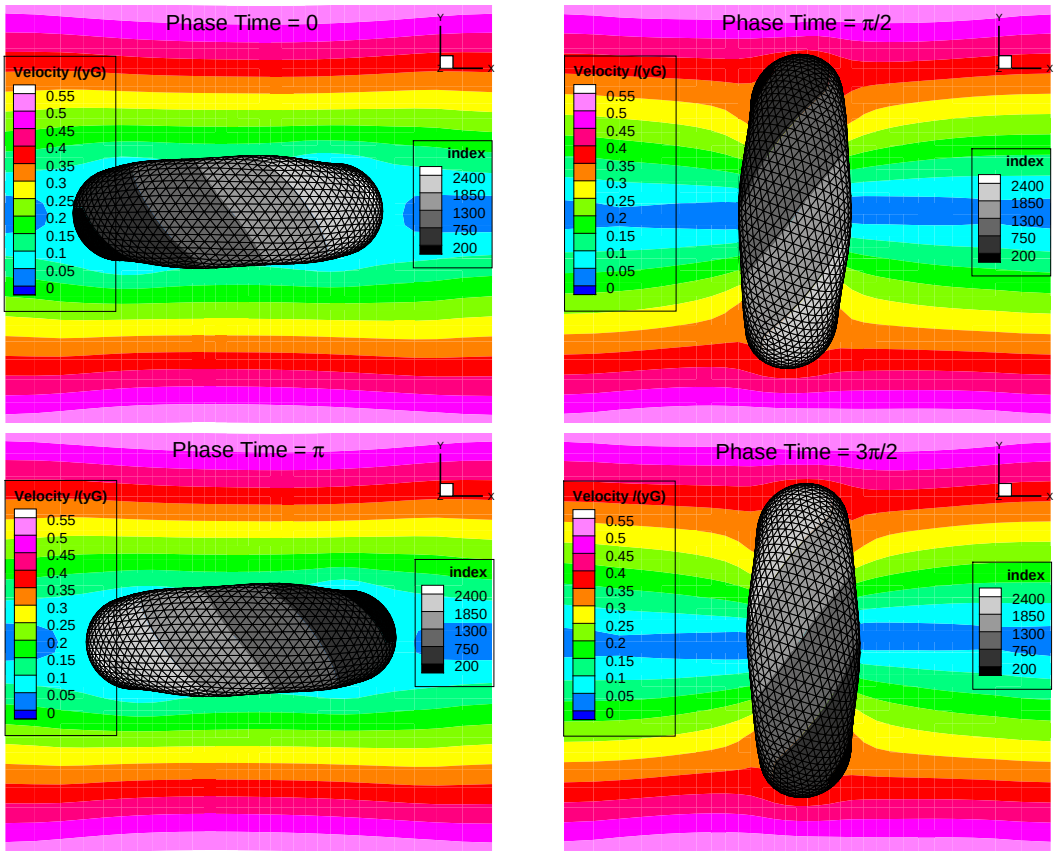


FIGURE 7.14: Snapshots of a RBC tumbling over a period where $G = 0.5$ s^{-1} and $\lambda_\mu = 0.27$. The normalised velocity magnitude of the suspending fluid is shown. The node index of the mesh is also shown to demonstrate the rotation of the RBC.

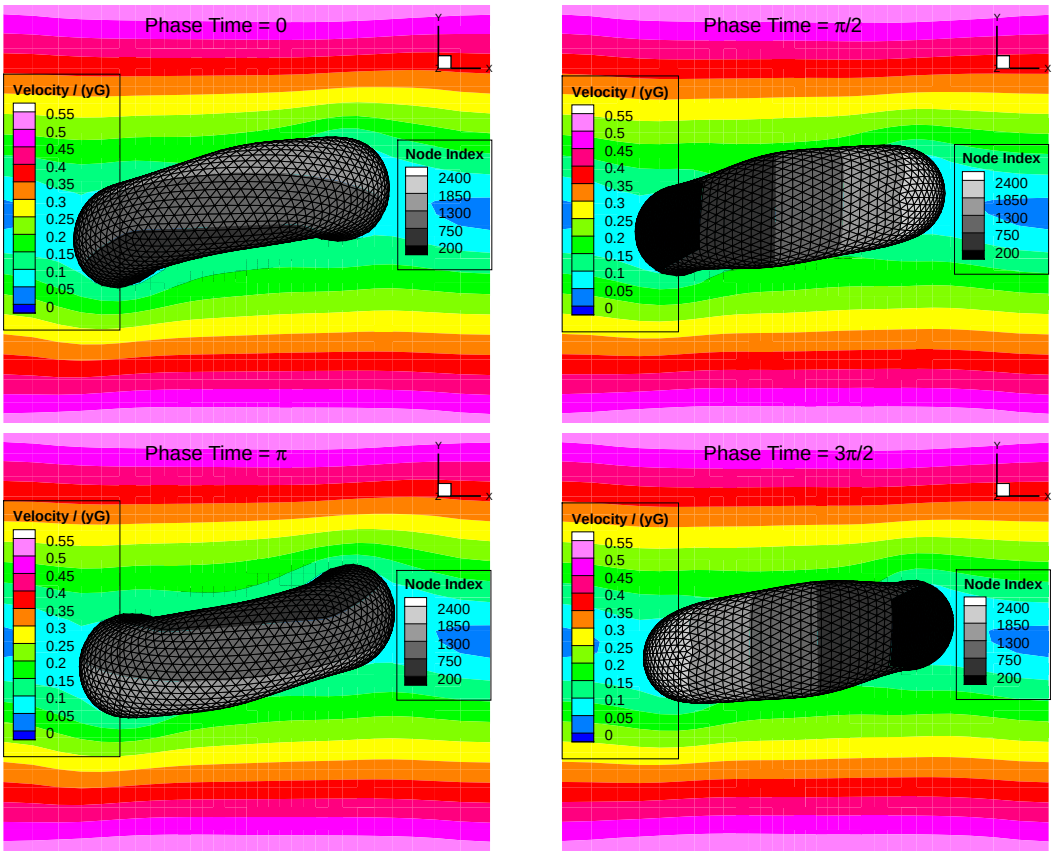


FIGURE 7.15: Snapshots of a RBC tank treading over a period where $G = 1.6\ \mathrm{s}^{-1}$ and $\lambda_\mu = 0.27$. The normalised velocity magnitude of the suspending fluid is shown. The node index of the mesh is also shown to demonstrate the rotation of the RBC.

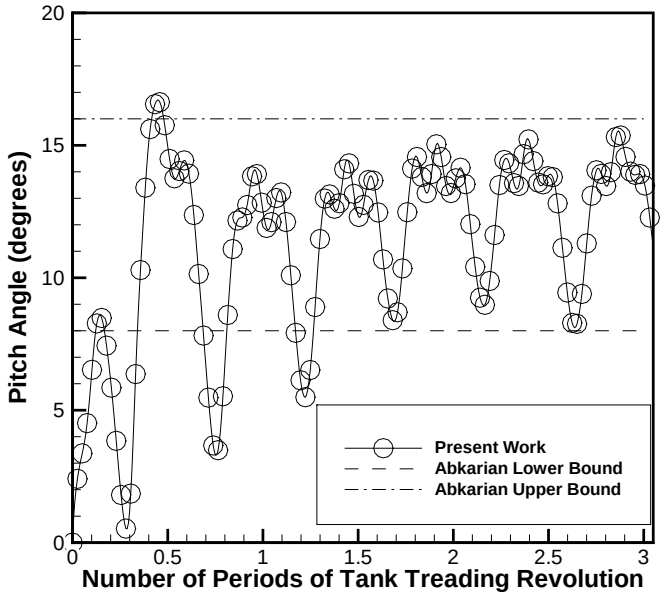


FIGURE 7.16: Pitch angle of a swinging RBC over multiple tank treading revolutions for $G = 1.8$ s^{-1} and $\lambda_\mu = 0.27$. Upper and lower bound of pitch angle from experimental measurements of Abkarian et al. is also shown [10].

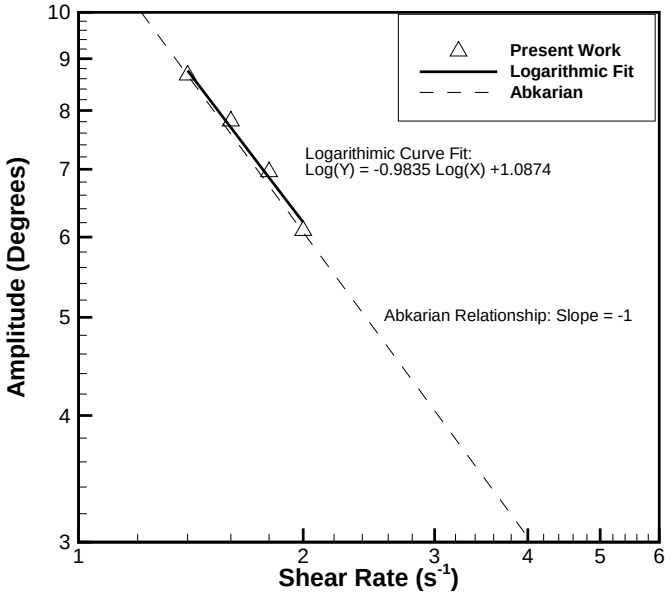


FIGURE 7.17: Amplitude of the swinging angle for a variety of shear rates. The linear relationship found by Abkarian et al. between the variables is also plotted.

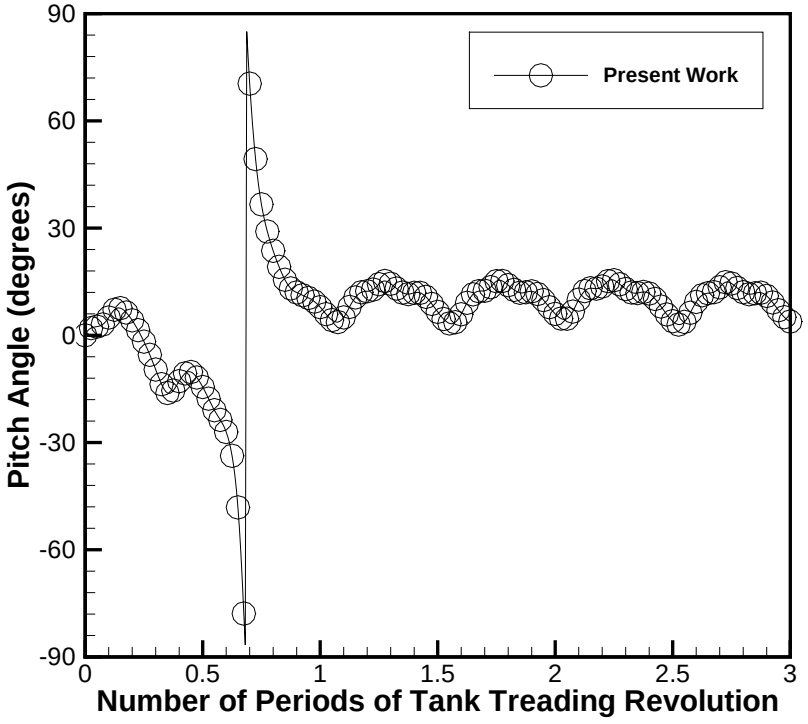


FIGURE 7.18: Pitch angle of RBC over multiple tank treading revolutions for $G = 1.2$ s^{-1} and $\lambda_\mu = 0.27$. The flow shows one single intermittent tumble.

7.5 Flow Problem Runtimes

A key aspect of modelling blood flow is the computational effort required to perform simulations. This is primarily due to the very small time step requirement of the ADE-SP LBFS. Initially the RK4 time integration scheme, as described in Section 2.4.2, was adopted for fluid flow. However the time step required to maintain stability for the flow problems in this chapter was an order of magnitude lower than that predicted by Equation (2.55). Subsequently a Forward Euler integration scheme was trialled and no adverse affect on stability was found for a large computational saving. This would suggest that the IBM and ADE-SP time integration is the limiting factor on stability criteria. This time step requirement lead to very large computation requirements due to the large number of iterations required to perform numerical simulations. The runtimes for each of the flow problems in this chapter are shown in Table 7.1. These runtimes confirm that the allowable time

Flow Problem	N_{Cells}	Δt (µs)	Iterations (10^6)	CPU time (s)	Simulated time (s)	Other
Optical Tweezers	64000	0.047	14.7	25897	0.69	$\mathbf{F} = 160$pN
Wheel	8000	0.1	4.87	11531	0.49	$G = 60\text{s}^{-1}$
Tank Treading	27000	3.125	12.0	15709	37.5	$G = 1.6\text{s}^{-1}$

TABLE 7.1: Runtimes of RBC flow problems simulated in this work. All simulations were run with a RBC with $N_{object\ nodes} = 2562$ and all simulations were performed on a Volta V100 GPU card.

step is the primary factor in determining the computation effort required. The timestep Δt is of the order of µs where as the flow problem requires time scales in the order of seconds to be simulated. This suggests that an alternate approach to time integration that allows larger time steps would offer large computational savings. It is also noticeable that the time per iteration is higher for the Wheel problem than the Tank Treading problem even though the Wheel problem has a lower N_{cells}. The increase in computational effort per iteration in the wheel problem is due to the VI-FS steps during the IBM process. This demonstrates the complex relationship between computational complexity and runtimes when GPUs are used. Depending on the orientation and position of the RBC this could lead to uncoallesced memory accesses and code divergence. The impact of these are discussed in further detail in Chapter 8.

7.6 Summary

This chapter has shown that the ADE-SP LBFS blood flow model implemented in this work is equivalent to that produced by Chen [1]. The optical tweezers numerical experiment performed produces equivalent results to Chen with the exact same RBC parameters. However when the ADE-SP LBFS blood flow model was validated against further experimental results, it was found that a pre-stressed biconcave mesh was required to simulate RBC dynamics with physiologically realistic values. A pre-stressed mesh using the values in Table 6.2 produces very accurate results for RBC dynamics in shear flow that agree with experimental evidence in the literature. However this produced more inaccurate results than the stress free configuration when the pre-stressed configuration was retrospectively used to simulate the optical tweezers and wheel deformation experiment. It resulted in larger deformations in the optical tweezers experiment and smaller deformations in the wheel experiment. This may be down to the spring model used. The modified WLC model proposed by Chen is stiffer than alternative spring models used in the literature. Fedosov [223] uses a WLC-POW model whereas Pivkin [72] and Reasor [99] use a WLC spring mixed with a hydrostatic elastic energy term. Another consideration is the choice of model for membrane viscosity. It is shown that the viscosity of Ehi-Egharevba [213] does not impact on RBC dynamics in blood flow scenarios. The work of Fedosov [73] shows that membrane viscosity has significant impact on RBC tank treading frequency and also swinging angle amplitudes. Membrane viscosity is captured in this work by assigning the membrane viscosity to the SPH particles in the fluid colocated with the membrane. This would suggest that to capture membrane viscosity in the current work, that the fluid volume's viscosity must be updated to reflect the viscosity of the membrane. This would require the modification of Equation (6.45). On the other hand the work of Pivkin [72] and Reasor [99] do not supply results for the amplitude of swinging angles in their work. It may be that if these models are adopted that they do not accurately capture the pitch orientations or swinging angles that are observed in experimental studies. Finally the runtimes of the simulations performed, confirm that the allowable time step is the primary factor in determining the computation effort required. The timestep Δt is of the order of µs where as the flow problem requires time scales in the order of seconds to be simulated. This would suggest that an alternate approach to time integration that allows larger time steps would offer large computational savings.

Chapter 8

General Purpose GPU Programming and Computational Optimisation

8.1 Introduction

As discussed in Section 1.8.2, it was decided to utilise GPGPU programming using the CUDA framework and Nvidia GPU cards to reduce runtimes this enabling physiologically realistic blood flow modelling in the future. This section outlines the design decisions that were made to optimise the runtime acceleration on GPGPUs and also the trade offs between applicable approaches to mesh topology. It will give a minimal overview of the CUDA architecture. If a more detailed understanding of the CUDA framework and GPU technology is required, please refer to the CUDA toolkit [235].

8.2 CUDA Framework

CUDA is a software library that allows programmers to interface with Nvidia GPU cards with programming languages including C++, C, Fortran and Python. It allows the programmer to focus on programming functionality rather than programming hardware. However a baseline knowledge of GPU hardware is required

to maximise the acceleration due to GPGPU programming. GPGPU programming essentially allows a common task to be performed in parallel by distributing each individual job to multiple cores in a GPU. For example if you wish to calculate the volume of every cell in a mesh, a CPU will calculate the volume of each cell sequentially whereas the GPU will calculate the volume of each cell in parallel. This is done by what's referred to as "calling a kernel" from your CPU language (C++ in this work). A kernel is essentially a function which tells your GPU to run this task in parallel using the specified amount of cores. It hides all interactions with the GPU and allows the programmer to focus on functionality rather than telling the GPU which core to run the task. An overview of this process is shown in Figure 8.1 for an example calculation. A streaming multiprocessor is a subset of cores on the GPU only accessible to that streaming multiprocessor. A block is a software abstraction and is a collection of warps. Warps are another software abstraction and contain 32 threads with common instructions, each thread is then run on a GPU core. Unified memory is a CUDA software abstraction which allows memory to be called from both CPU and GPU. The key point for a programmer to take from this overview is that CUDA will split the task into parcels of work called warps which contain 32 threads. Each thread performs one of the parallel jobs to be performed. Key to note is that each warp performs the same operations on each of these 32 threads. This will be discussed in more detail later in this chapter.

8.3 Compute Capability

Nvidia GPUs were used in this work. A Tesla K40c was kindly donated by Nvidia Corporation. This GPU was used on a desktop basis for development, debugging and profiling. Towards the end of this work the Irish supercomputer Kay hosted by ICHEC was used to run simulations. This allowed the author to run simulations on Tesla V100 GPUs. This was done for running simulations after concepts were proven locally on the Tesla K40c. A comparison of the compute capability of the Volta V100 relative to previous generations of Nvidia GPUs is shown in Figure 8.2.

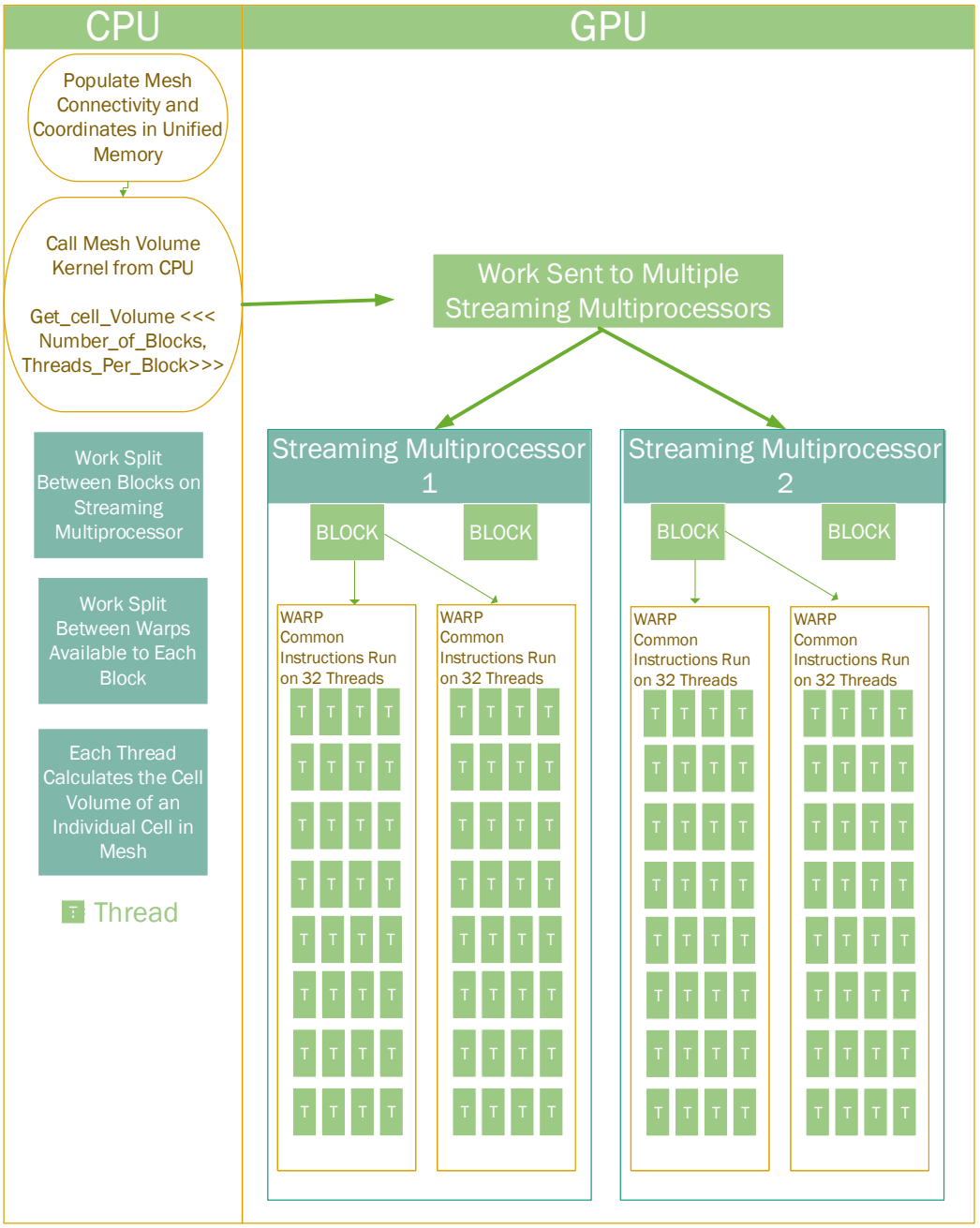


FIGURE 8.1: Workflow of CUDA kernel - shows declaration of memory in Unified Memory, kernel calling and Streaming Multiprocessor architecture for cell volume calculation.

8.4 Programming Best Practice

The best practice for CUDA programming [236] recommends the following approach to GPGPU development:

Table 1. Comparison of NVIDIA Tesla GPUs

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1530 MHz
Peak FP32 TFLOPS[1]	5	6.8	10.6	15.7
Peak FP64 TFLOPS[1]	1.7	.21	5.3	7.8
Peak Tensor TFLOPS[1]	NA	NA	NA	125
Texture Units	240	192	224	320
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB	6144 KB
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB	Configurable up to 96 KB
Register File Size / SM	256 KB	256 KB	256 KB	256KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB	20480 KB
TDP	235 Watts	250 Watts	300 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion	21.1 billion
GPU Die Size	551 mm^2	601 mm^2	610 mm^2	815 mm^2
Manufacturing Process	28 nm	28 nm	16 nm FinFET+	12 nm FFN
[1] Peak TFLOPS rates are based on GPU Boost Clock				

FIGURE 8.2: Compute capability of the Volta V100 relative to previous generations of Nvidia GPUs. Extracted from Nvidia Volta white paper [11].

- Profile Code.
- Parallelise.
- Optimise.
- Deploy.

Code development is a timely and costly process. There is no point in spending valuable time improving code that accounts for less than one percent of runtime of the overall program. To identify the key bottlenecks in the program, it is recommended to use a profiling tool such as GPROF. This identifies the key functions that should be parallelised. After these functions are parallelised, it is recommended to use NVPROF, the Nvidia CUDA profiler, to measure the speed up in these functions. If certain functions are still highly time consuming, it is recommended to spend time optimising them with the following high priority design implementations:

- Develop approaches for parallelisation of sequential code.
- Minimise data transfers between the CPU and the GPU.
- Adjust kernel launch configuration to maximize GPU utilisation.
- Ensure global memory accesses are coalesced.
- Minimise redundant accesses to global memory whenever possible.
- Avoid long sequences of diverged execution by threads within the same warp.
- Avoid race conditions.

These are now described in turn.

8.4.1 Find ways to parallelise sequential code

In the example in Section 8.2, the cell was used as a base unit of parallelisation. Impediments to parallelisation include the cell requiring prerequisite information about neighbouring cells to complete a calculation e.g. neighbouring cell velocities for calculating the velocity gradient in the current cell. Reordering kernels so that the prerequisite information is calculated in parallel before the main cell volume calculations can avoid the need for sequential code. Another key decision is choosing the base unit of, for example, flux calculations; do you calculate the flux in and out of a cell volume simultaneously or calculate the flux at each face simultaneously? This decision has many impacts and will be discussed in more detail later as it depends highly on the solver used.

8.4.2 Minimise data transfers between the CPU and the GPU

On many occasions there is information which is stored in memory on the CPU which is required on the GPU to perform calculations. However there is a limit in bandwidth capacity for transferring data from CPU to GPU and vice versa (see Figure 8.3). As a result data transfers are very time consuming processes and should be minimised. In blood flow numerics, it is possible to run the solver completely on the GPU without CPU interactions. The only prerequisite information

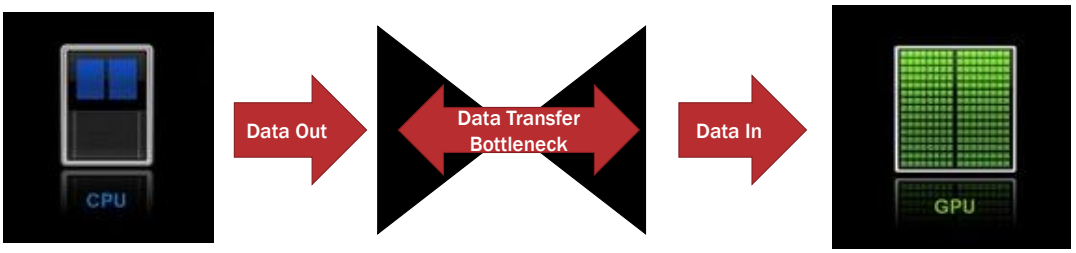


FIGURE 8.3: Illustration of bandwith bottleneck in data transfers from CPU to GPU.

includes mesh details and initial conditions. This can be calculated once and sent to the solver before the initial time step. The only remaining need for a data transfer is for postprocessing source data, *i.e* numerical solution values during runtimes, as they cannot be written in Tecplot formats from the GPU directly. This is a time consuming process and it is recommended to export postprocessing data at an arbitrary frequency, e.g. every 10000 iterations. This frequency is dependent on the accuracy required by the time scale of the flow problem. Increasing the frequency will give more accurate postprocessing but will increase runtimes.

8.4.3 Adjust kernel launch configuration to maximize GPU utilisation

When calling a CUDA kernel, it is possible to specify the number of threads per block. This is a key consideration when trying to ensure that the GPU device is as busy as possible. Key recommendations are to use the CUDA occupancy calculator to optimise the occupancy, *i.e.* percentage of CUDA cores active in GPU at a given time, and to keep the number of threads per block to multiples of 32.

8.4.4 Ensure global memory accesses are coalesced

Most data on a GPU is stored in global memory and is then transferred to warps as required. Similar to CPU and GPU memory, this data transfer takes time and should be minimised as much as possible. When accessing memory, the warp accesses memory in 32 byte memory transactions. When 8 byte double precision variables are used, this results in 4 variables per transaction. To ensure maximum

efficiency each of those 4 variables should be used by the warp in calculations. To achieve this efficiency in blood flow numerics, it is recommended to use structured uniform or non-uniform grids. Structured grids store variables in sequential order and have structured connectivity rules which ensure maximum efficiency of global memory accesses. Unstructured grids on the other hand have random connectivity relationships. This leads to more memory transactions to access all the global memory required by a warp. This trade off in mesh choice is discussed in more detail later.

8.4.5 Minimise redundant accesses to global memory whenever possible

In addition to the previous recommendations on global memory accesses, where it is identified that global memory accesses are time consuming operations, it is recommended that the number of global memory accesses be reduced as much as possible. An example of this may be a property that is calculated as the product of two constant global memory variables. Rather than two global memory accesses, it is recommended to calculate this new property once at the start of the code and store the property in memory. This reduces the global memory accesses from two to one. Secondly, it is important to ensure that kernels only call the global memory variables it needs to calculate the output and no more.

8.4.6 Avoid long sequences of diverged execution by threads within the same warp

As shown in Figure 8.1, each warp executes the same set of instructions on 32 threads. This means that if there are decision trees or loops within the kernel, they should have the same outcome for every thread within a warp. If there is a divergence, the warp will perform the instructions for every thread once for one decision path and then perform the instructions again for the second decision path. If loops within a warp have different numbers of iterations, every thread in the warp will perform at the speed of the largest loop. In blood flow numerics, this issue is primarily introduced by having meshes with different cell types *i.e* tetrahedrons and hexahedrons.

8.4.7 Avoid race conditions

A race condition occurs when two threads are trying to write memory to the same memory location. This occurs in many cases in numerics, e.g. summing fluxes in a cell volume. As threads are fully parallel there is a chance that both threads will finish their computation at the same time and update global memory at the same time. This results in an incorrect update of the flux in the cell volume due to the race condition (see Figure 8.4). The solution to this problem is to use what

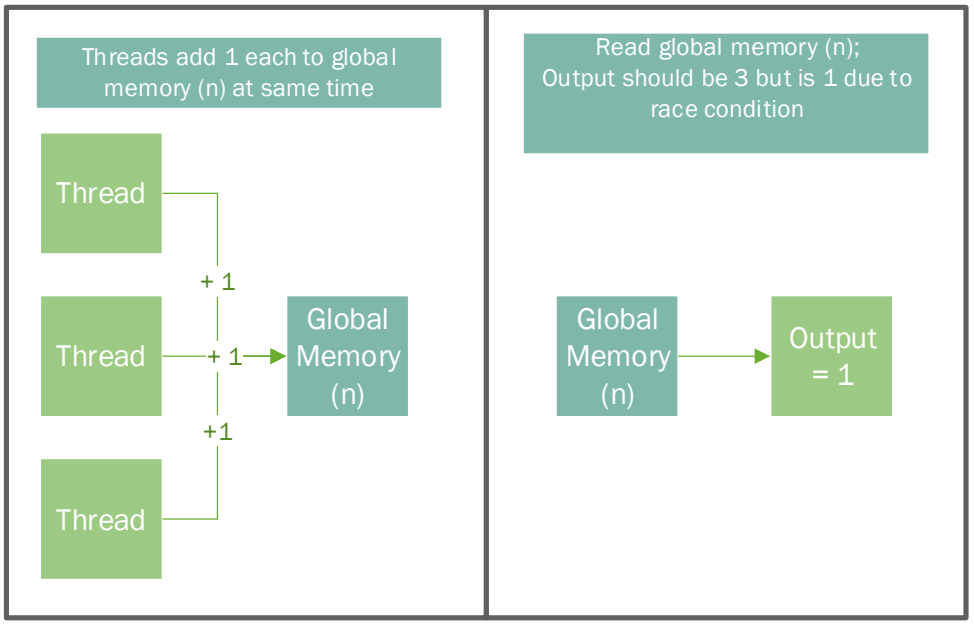


FIGURE 8.4: Illustration of race condition.

is called atomic operations. These ensure race conditions are avoided but come at a computational expense as it means that instructions are executed sequentially rather than in a parallel fashion. As GPUs are slower than CPUs at sequential operations, this can result in lengthy bottlenecks.

8.5 Design Decisions and GPUs

In the previous sections, CUDA architecture and some key design considerations for GPGPU development were introduced. This section will show how they were

considered in the development of the blood flow solver and provided recommendations for future developers.

8.5.1 LBFS - Fluid Solver

In the LBFS, there are four main functions that have to be performed:

- Calculate local time step.
- Calculate gradients at cell centres.
- Calculate fluxes at each face.
- Time integration of N-S equations.

After profiling of the CPU code it was found that flux calculations accounted for ≥ 80 percent of the runtime and the flux calculations were ported to GPU code initially. After this port, profiling the GPU code found that the vast majority of time was spent transferring the solution of the conserved variables from GPU to CPU and vice versa. To avoid these GPU-CPU data transfers, all four steps were ported to GPU code with base units as described in Table 8.1. After profiling

Function	Base unit of parallelisation
Time step	Cell volume
Gradients	Cell volume
Fluxes	Cell faces
Time integration	Cell volume

TABLE 8.1: Base unit of parallelisation for LBFS functions.

the new fully GPU ported code, the flux calculations still accounted for ≥ 80 percent of the runtime. As per best practice, the flux calculation function was then subjected to more intense profiling and the function was thoroughly investigated for opportunities to improve runtimes as per Section 8.4. The outcomes of the investigation resulted in three key decisions to make: choice of base unit of parallelisation, availability of modern GPU hardware and mesh choice.

8.5.1.1 Flux calculations - base unit of parallelisation

The choice of base unit of parallelisation for the flux calculations has a large impact on runtimes. Depending on the choice of base unit, it will either result in race conditions and use of atomics or double the calculations required. When calculating fluxes in and out of the cell volumes, the flux at each face is calculated and summed together with the fluxes at each face in the cell. If the base unit of parallelisation is chosen as the face, this results in race conditions and the necessary use of atomics to add fluxes together for each cell. The alternative is to use cell volumes as a base unit. To avoid atomics with this approach requires the computation of the flux at each cell face twice, once for each neighbouring cell. Using this approach can also introduce code divergence if meshes with multiple cell types or hanging nodes are used.

8.5.1.2 Availability of modern GPU hardware

When profiling the flux calculations on the GPUs, it was discovered that the atomic operations necessary for calculating the cell fluxes were responsible for 60 percent of the runtime on a Tesla K40. Using the same code on a modern Tesla V100 resulted in 5× reduction in runtimes. This is mainly due to the significant improvements in atomic operations in modern GPU hardware.

8.5.1.3 Mesh choice

The use of structured or unstructured meshes has the largest impact on runtimes. As discussed before there is a benefit from coalesced global memory accesses using structured grids. This is common to all solvers. However there are some specific traits of the LBFS that make structured meshes computationally cheaper than unstructured meshes. First off structured meshes allow a reduction in the number of calculations required at each face. Due to the face normal having two zero terms, this makes a lot of calculations redundant and can be removed from the code. For example, this means that only 4 terms are required to be calculated in the flux tensor in Equation (2.2) as opposed to 10 terms with an unstructured grid. Another key aspect of the face calculation is the interpolation of the macroscopic variables (see Equation (2.14)). As part of the approach it is required to establish which cell neighbour the lattice nodes lie in. As shown in listing 8.1, there are

3 diverging conditions for every two lattice velocities. Using D3Q15, this could result in $3 \times 7 = 21$ diverging conditions across 32 threads in a warp. As each warp performs common instructions on each thread, this could result in a $21\times$ increase in runtimes. This was a main motivation behind using hexahedral meshes as they tend to be more consistent in aspect ratio and normals than tetrahedral meshes. Using any structured mesh removes any divergence in the GPU code.

```
///   case 5 and 6: // back node and front node

if (cell_normal.z > d_1) {

        rho_lattice[5] = rho_i - grad_rho_1.z* dt;
        u_lattice[5] = u_i - grad_u_1.z* dt;
        v_lattice[5] = v_i - grad_v_1.z *dt;
        w_lattice[5] = w_i - grad_w_1.z *dt;

        rho_lattice[6] = rho_nb + grad_rho_2.z* dt;
        u_lattice[6] = u_nb + grad_u_2.z* dt;
        v_lattice[6] = v_nb + grad_v_2.z *dt;
        w_lattice[6] = w_nb + grad_w_2.z *dt;

}
else if (cell_normal.z < -d_1) {
        rho_lattice[5] = rho_nb - grad_rho_2.z* dt;
        u_lattice[5] = u_nb - grad_u_2.z* dt;
        v_lattice[5] = v_nb - grad_v_2.z *dt;
        w_lattice[5] = w_nb - grad_w_2.z *dt;

        rho_lattice[6] = rho_i + grad_rho_1.z* dt;
        u_lattice[6] = u_i + grad_u_1.z* dt;
        v_lattice[6] = v_i + grad_v_1.z *dt;
        w_lattice[6] = w_i + grad_w_1.z *dt;
}
else {
        rho_lattice[5] = rho_lattice[0] - grad_rho_3.z* dt;
        u_lattice[5] = u_lattice[0] - grad_u_3.z* dt;
        v_lattice[5] = v_lattice[0] - grad_v_3.z *dt;
        w_lattice[5] = w_lattice[0] - grad_w_3.z *dt;

        rho_lattice[6] = rho_lattice[0] + grad_rho_3.z* dt;
        u_lattice[6] = u_lattice[0] + grad_u_3.z* dt;
        v_lattice[6] = v_lattice[0] + grad_v_3.z *dt;
        w_lattice[6] = w_lattice[0] + grad_w_3.z *dt;

}
```

Listing 8.1: Interpolation Divergence

8.5.1.4 Conclusion

To summarise it is recommended to use cell faces as a base unit for flux calculations if there is access to modern Tesla V100 hardware. It is also recommended where possible to use a structured grid be it uniform or non-uniform as both reduce the amount of calculations required and reduce the divergence of threads during flux calculations.

8.5.2 IBM

In the IBM, there are four main functions that have to be performed:

- Velocity interpolation on nodes.
- Update of node positions.
- Update node forces (RBC solver in next section).
- Force spreading.

Similar to the LBFS porting approach, all four functions were ported to GPUs as they used data which was already being stored on global memory in GPUs. This avoided unnecessary transfer of data from GPU to CPU. The base units for each function are described in Table 8.2. The main design decision here is to decide

Function	Base unit of parallelisation
Velocity interpolation	Object nodes
Update of node positions	Object nodes
Force spreading	Object nodes

TABLE 8.2: Base unit of parallelisation for IBM functions.

on the base unit of the force spreading function. If a base unit of object nodes is adopted it results in the necessary use of atomics to add restoring forces to each cell volume. If a base unit of cell volumes is adopted, it results in excessive computations. The former results in $O(N_{object\ nodes} \times 64)$ atomic operations whereas the latter results in $O(N_{cells} \times N_{object\ nodes})$ parallel operations. For a test case involving $N_{object\ nodes} = 2562$ and $N_{cells} = 64000$ this results in $10000\times$ less operations for the former approach. As this work was performed for single red blood

cells, this dynamic may change when additional red blood cells are added to the flow and should be investigated for computational efficiency.

8.5.3 RBC Structural Model

In the RBC model there are two functions to consider:

- Internal RBC viscosity assignment.
- RBC force calculation.

Similar to the LBFS and IBM porting approach, all functions were ported to GPUs as they used data which was already being stored on global memory in GPUs. This avoided unnecessary transfer of data from GPU to CPU. The base units for each function are described in Table 8.3. The bulk of runtime is spent in

Function	Base unit of parallelisation
Viscosity	Object nodes
RBC Forces	Object springs

Table 8.3: Base unit of parallelisation for IBM functions.

the RBC forces function. There are two options here for base units: to use object nodes or object springs. Using object nodes has the benefit of avoiding atomics but involves a far greater number of global memory accesses as node connectivity is required in the RBC kernel. It also involves twice the number of calculations as the force from each spring is calculated for each spring node. The use of object springs results in less global memory accesses and calculations are performed only once for each spring, but it does involve the use of atomics to sum the forces on each object node. The former approach performs better on Tesla K40 whereas the latter performs better on the Volta V100.

8.6 Indicative Runtimes

To demonstrate the benefit of using GPGPU enabled hardware a variety of problems were profiled and the CPU clock times were recorded. Identical simulations

were profiled on CPU (where applicable), a Tesla K40 and a Volta V100. The CPU simulations were performed on a AMD Fx-6300 chip. Each problem was run for 25000 iterations and the runtime was recorded. Three runs of each problem were performed and the average time of the three runs was taken as the benchmark time. A comparison of the runtimes for a fluid only problem is shown in Table 8.4. A comparison of runtimes for a ADE-SP RBC solver only problem is shown in Table 8.5. Finally, a comparison of the ADE-SP LBFS blood flow solver is done for the Tesla K40 and a Volta V100 only as a CPU version is not available. The results for this benchmark are shown in Table 8.6. It is worth noting that significant improvements in runtimes are made by using GPUs versus CPUs.

Test Case	CPU	Tesla K40	Volta V100
Runtime (s): Lid Driven Cavity - Structured Mesh	15550	317	129
Runtime (s): Lid Driven Cavity - Unstructured Mesh	18401	930	392
Runtime Ratio: Structured Mesh	120.5	2.5	1.0
Runtime Ratio: Unstructured	142.6	7.2	3.0

TABLE 8.4: Indicative runtimes in seconds and runtime ratios of simulations performed on GPUs and CPU with the LBFS for fluid flow problem only with 1000000 cells in the fluid domain.

Test Case	CPU	Tesla K40	Volta V100
Runtime (s): RBC - Optical Tweezers (No Fluid)	483	20	12
Runtime Ratio: RBC	40.3	1.7	1.0

TABLE 8.5: Indicative runtimes in seconds and runtime ratios of simulations performed on GPUs and CPU with the ADE-SP RBC solver with 2562 object nodes on the RBC mesh.

Test Case	CPU	Tesla K40	Volta V100
Runtime (s): RBC - Optical Tweezers (Fluid Present)	-	126	28
Runtime Ratio:	-	4.5	1.0

TABLE 8.6: Indicative runtimes in seconds and runtime ratios of simulations performed on GPUs with the full ADE-SP LBFS blood flow solver with 2562 object nodes on the RBC mesh and 64000 cells in the fluid domain.

8.7 Summary

In conclusion the use of GPGPU programming enables large acceleration in runtimes for the current ADE-SP LBFS blood flow model. The extent of the acceleration achieved is highly dependent on two main user choices: use of structured or unstructured meshes and choice of GPU hardware. The use of structured grids enables significant speed ups over fully unstructured grids due to a reduction in calculations and increased memory coalescence. Modern GPU hardware also offers significant improvements over older hardware due to extensive improvements in the speed of atomic operations in modern hardware.

Chapter 9

Conclusions and Recommendations

9.1 Conclusions

In this work a full ADE-SP LBFS blood flow model has been developed, and verified and validated against a variety of experimental and numerical results in the literature. In Chapter 4 it was demonstrated that the LBFS is capable of predicting solutions to the N-S equations in 3D using fully unstructured hexahedral grids. It was shown that utilising preconditioning can significantly accelerate runtimes of the LBFS when simulating convection dominated steady flow. Preconditioning also offers increased levels of accuracy when utilised on fully unstructured grids with a large variation in cell size. In Chapter 5 an IBM that can handle non-uniform grids is succesfully verified against benchmark results in the literature. This work was a prerequisite for coupling a ADE-SP RBC model with a N-S solver.

In Chapter 7 it was demonstrated that the ADE-SP LBFS blood flow model can fully capture the complex RBC dynamics of tank treading, swinging and tumbling. During these experiments, it was found that a prestressed biconcave mesh was required to simulate RBC dynamics with physiologically realistic values of RBC elastic properties. A prestressed mesh using the values in Table 6.2 produces very accurate results for RBC dynamics in shear flow that agree with experimental measurements in the literature. However this produced more inaccurate results than

the stress free configuration when the prestressed configuration was retrospectively used to simulate the optical tweezers and wheel deformation experiment. This may be down to the spring model used. The modified WLC model proposed by Chen is significantly stiffer than alternative spring models used in the literature. Fedosov [223] uses a WLC-POW model whereas Pivkin [72] and Reasor [99] use a WLC spring mixed with a hydrostatic elastic energy term. Another consideration is the choice of model for membrane viscosity. It was discovered that the model of Ehi-Egharevba [213] does not impact on RBC dynamics in blood flow scenarios. The work of Fedosov [73] shows that membrane viscosity has significant impact on RBC tank treading frequency and also swinging angle amplitudes. Membrane viscosity is captured in this work by assigning the membrane viscosity to the SPH particles in the fluid colocated with the membrane. This would suggest that to capture membrane viscosity in the current work, that the fluid volume's viscosity must be updated to reflect the viscosity of the membrane. This would require the modification of Equation (6.45). On the other hand the work of Pivkin [72] and Reasor [99] do not supply results for the amplitude of swinging angles in their work. It may be that if these models are adopted that they do not accurately capture the pitch orientations or swinging angles that are observed in experimental studies.

Finally the ADE-SP LBFS blood flow model was accelerated using a GPGPU which offers significant performance improvements over CPU solvers. However the extent of the acceleration achieved is highly dependent on two main user choices: use of structured or unstructured meshes and choice of GPU hardware. The use of structured grids enables significant speed ups over fully unstructured grids due to a reduction in calculations and increased memory coalescence. Modern GPU hardware also offers significant improvements over older hardware due to extensive improvements in the speed of atomic operations with modern hardware.

9.2 Recommendations for Future Work

The scope of the project was to investigate the outcome of coupling the ADE-SP RBC model with a N-S solver and predicting RBC dynamics in flow. The investigations performed in this work show that there is a lack of a uniform configuration of the RBC that satisfies all experimental results in the literature. Due to the lack of results in the literature it is hard to ascertain if this is due to the choice of spring

model or that all spring models are incapable of fully capturing all experimental results. There is also an element of uncertainty in the literature with regards to the correct stress free shape of the RBC for SP models. Therefore the following future investigation is proposed:

- Investigation of RBC dynamics for the modified WLC model (current work), the WLC-POW model and the WLC model with hydrostatic elastic energy constant.
- Comparison of equilibrium shape for continuum and spring-particle methods. In particular investigate compressive and tensile stresses in the biconcave shape.

As mentioned previously, the choice of membrane viscosity model has no impact on the results of the RBC dynamics simulations. A different approach that updates the cell volumes of the fluid to reflect the RBC membrane could have significant impact on RBC dynamics and would be a worthwhile investigation.

Once the aforementioned investigations have been performed the ADE-SP RBC model could be then extended to a variety of other blood flow problems including parachute flow, flow through a micro-channel, aggregation, and dynamics of complex morphologies in shear flow. The possiblities are endless.

As shown in Chapter 8 and Chapter 5, utilising non uniform grids can offer significant savings in computational expense over fully unstructured flows. However this was not benchmarked in this work and it is recommended that a fully unstructured IBM be developed and implemented. This should then be benchmarked against the current work.

It is shown in Chapter 8 that the use of structured meshes offers significant acceleration over unstructured meshes when utilising GPUs. However in this work tetrahedral meshes are used to discretise the RBC which perform poorly on GPUs with regards to memory coalescence. It would be an interesting study to investigate the performance of the ADE-SP RBC model on GPUs when hexahedral meshes are employed. Hexahedral meshes tend to offer better memory coalescence and could offer significant savings in runtimes.

Finally one of the largest impacts on computational costs is the very low time step required during RBC dynamics. It is proposed that a similar approach to

that used by Breen et al. [237] to accelerate the *many body problem* could offer significant savings in runtimes. In their work, a deep neural network is trained on the results of a classical numerical solver and subsequently can predict the results of the classical solver up to 100 million times faster over a bounded time interval.

Appendix A

Governing Equation Aids

A.1 Introduction

This appendix includes aids to understanding the governing equations of the LBFS described in Chapter 2. The aids include: an explicit D3Q15 Discretisation of N-S Equations in the LBFS, a full discussion of non-dimensional form of the LBFS, the equations of the Hybrid Green-Gauss/Weighted-Least-Squares Gradient Operator, an explicit writing of non-conservative form of N-S equations and computer code for calculating the RK4 stability region.

A.2 Explicit D3Q15 Discretisation of N-S Equations

This appendix contains an explicit description of how the mass and momentum fluxes are calculated using the LBFS. These fluxes are then used in the N-S equation to calculate the change in macroscopic variables from time step to time step. The flux tensor in Equation (2.2) can be re-written as:

$$\mathbf{F} = Flux\ Tensor = \begin{pmatrix} P_x & P_y & P_z \\ \Pi_{xx} & \Pi_{xy} & \Pi_{xz} \\ \Pi_{yx} & \Pi_{yy} & \Pi_{yz} \\ \Pi_{zx} & \Pi_{zy} & \Pi_{zz} \end{pmatrix} \tag{A.1}$$

where P is the mass flux, Π is the momentum flux and x, y, z indicate the Cartesian axis of the flux contribution. Using Equation (2.4) and Table 2.1 the individual parts of the flux tensor can be written explicitly in terms of lattice velocities, equilibrium distribution functions and non-equilibrium functions as follows:

$$\begin{aligned}
P_x =& e_{1,x} f_1^{eq}(\mathbf{r},t) + e_{2,x} f_2^{eq}(\mathbf{r},t) + e_{7,x} f_7^{eq}(\mathbf{r},t) \\
&+ e_{8,x} f_8^{eq}(\mathbf{r},t) + e_{9,x} f_9^{eq}(\mathbf{r},t) + e_{10,x} f_{10}^{eq}(\mathbf{r},t) \\
&+ e_{11,x} f_{11}^{eq}(\mathbf{r},t) + e_{12,x} f_{12}^{eq}(\mathbf{r},t) + e_{13,x} f_{13}^{eq}(\mathbf{r},t) \\
&+ e_{14,x} f_{14}^{eq}(\mathbf{r},t) \\
=& (1) f_1^{eq}(\mathbf{r},t) + (-1) f_2^{eq}(\mathbf{r},t) + (1) f_7^{eq}(\mathbf{r},t) \\
&+ (-1) f_8^{eq}(\mathbf{r},t) + (1) f_9^{eq}(\mathbf{r},t) + (-1) f_{10}^{eq}(\mathbf{r},t) \\
&+ (1) f_{11}^{eq}(\mathbf{r},t) + (-1) f_{12}^{eq}(\mathbf{r},t) + (-1) f_{13}^{eq}(\mathbf{r},t) \\
&+ (1) f_{14}^{eq}(\mathbf{r},t)
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
P_y =& e_{3,y} f_3^{eq}(\mathbf{r},t) + e_{4,y} f_4^{eq}(\mathbf{r},t) + e_{7,y} f_7^{eq}(\mathbf{r},t) \\
&+ e_{8,y} f_8^{eq}(\mathbf{r},t) + e_{9,y} f_9^{eq}(\mathbf{r},t) + e_{10,y} f_{10}^{eq}(\mathbf{r},t) \\
&+ e_{11,y} f_{11}^{eq}(\mathbf{r},t) + e_{12,y} f_{12}^{eq}(\mathbf{r},t) + e_{13,y} f_{13}^{eq}(\mathbf{r},t) \\
&+ e_{14,y} f_{14}^{eq}(\mathbf{r},t) \\
=& (1) f_3^{eq}(\mathbf{r},t) + (-1) f_4^{eq}(\mathbf{r},t) + (1) f_7^{eq}(\mathbf{r},t) \\
&+ (-1) f_8^{eq}(\mathbf{r},t) + (1) f_9^{eq}(\mathbf{r},t) + (-1) f_{10}^{eq}(\mathbf{r},t) \\
&+ (-1) f_{11}^{eq}(\mathbf{r},t) + (1) f_{12}^{eq}(\mathbf{r},t) + (1) f_{13}^{eq}(\mathbf{r},t) \\
&+ (-1) f_{14}^{eq}(\mathbf{r},t)
\end{aligned} \tag{A.3}$$

$$
\begin{aligned}
P_z =& e_{5,z} f_5^{eq}(\mathbf{r},t) + e_{6,z} f_6^{eq}(\mathbf{r},t) + e_{7,z} f_7^{eq}(\mathbf{r},t) \\
&+ e_{8,z} f_8^{eq}(\mathbf{r},t) + e_{9,z} f_9^{eq}(\mathbf{r},t) + e_{10,z} f_{10}^{eq}(\mathbf{r},t) \\
&+ e_{11,z} f_{11}^{eq}(\mathbf{r},t) + e_{12,z} f_{12}^{eq}(\mathbf{r},t) + e_{13,z} f_{13}^{eq}(\mathbf{r},t) \\
&+ e_{14,z} f_{14}^{eq}(\mathbf{r},t) \\
=& (1) f_5^{eq}(\mathbf{r},t) + (-1) f_6^{eq}(\mathbf{r},t) + (1) f_7^{eq}(\mathbf{r},t) \\
&+ (-1) f_8^{eq}(\mathbf{r},t) + (-1) f_9^{eq}(\mathbf{r},t) + (1) f_{10}^{eq}(\mathbf{r},t) \\
&+ (1) f_{11}^{eq}(\mathbf{r},t) + (-1) f_{12}^{eq}(\mathbf{r},t) + (1) f_{13}^{eq}(\mathbf{r},t) \\
&+ (-1) f_{14}^{eq}(\mathbf{r},t)
\end{aligned}
\tag{A.4}
$$

$$
\begin{aligned}
\Pi_{xx} =& e_{1,x}^2 \left[f_1^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_1^{neq}(\mathbf{r},t) \right] + e_{2,x}^2 \left[f_2^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_2^{neq}(\mathbf{r},t) \right] \\
&+ e_{7,x}^2 \left[f_7^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_7^{neq}(\mathbf{r},t) \right] + e_{8,x}^2 \left[f_8^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_8^{neq}(\mathbf{r},t) \right] \\
&+ e_{9,x}^2 \left[f_9^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_9^{neq}(\mathbf{r},t) \right] + e_{10,x}^2 \left[f_{10}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{10}^{neq}(\mathbf{r},t) \right] \\
&+ e_{11,x}^2 \left[f_{11}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{11}^{neq}(\mathbf{r},t) \right] + e_{12,x}^2 \left[f_{12}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{12}^{neq}(\mathbf{r},t) \right] \\
&+ e_{13,x}^2 \left[f_{13}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{13}^{neq}(\mathbf{r},t) \right] + e_{14,x}^2 \left[f_{14}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{14}^{neq}(\mathbf{r},t) \right] \\
=& (1) \left[f_1^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_1^{neq}(\mathbf{r},t) \right] + (1) \left[f_3^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_3^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_7^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_7^{neq}(\mathbf{r},t) \right] + (1) \left[f_8^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_8^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_9^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_9^{neq}(\mathbf{r},t) \right] + (1) \left[f_{10}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{10}^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_{11}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{11}^{neq}(\mathbf{r},t) \right] + (1) \left[f_{12}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{12}^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_{13}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{13}^{neq}(\mathbf{r},t) \right] + (1) \left[f_{14}^{eq}(\mathbf{r},t) + \left(1 - \frac{1}{2\tau}\right) f_{14}^{neq}(\mathbf{r},t) \right]
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
\Pi_{yy} =& e_{3,y}^2 \left[f_3^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_3^{neq}(\mathbf{r},t) \right] + e_{4,y}^2 \left[f_4^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_4^{neq}(\mathbf{r},t) \right] \\
&+ e_{7,y}^2 \left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_7^{neq}(\mathbf{r},t) \right] + e_{8,y}^2 \left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_8^{neq}(\mathbf{r},t) \right] \\
&+ e_{9,y}^2 \left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_9^{neq}(\mathbf{r},t) \right] + e_{10,y}^2 \left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{10}^{neq}(\mathbf{r},t) \right] \\
&+ e_{11,y}^2 \left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{11}^{neq}(\mathbf{r},t) \right] + e_{12,y}^2 \left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{12}^{neq}(\mathbf{r},t) \right] \\
&+ e_{13,y}^2 \left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{13}^{neq}(\mathbf{r},t) \right] + e_{14,y}^2 \left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{14}^{neq}(\mathbf{r},t) \right] \\
\\
&= (1) \left[f_1^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_1^{neq}(\mathbf{r},t) \right] + (1) \left[f_3^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_3^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_7^{neq}(\mathbf{r},t) \right] + (1) \left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_8^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_9^{neq}(\mathbf{r},t) \right] + (1) \left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{10}^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{11}^{neq}(\mathbf{r},t) \right] + (1) \left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{12}^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{13}^{neq}(\mathbf{r},t) \right] + (1) \left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{14}^{neq}(\mathbf{r},t) \right]
\end{aligned}
\tag{A.6}
$$

$$
\begin{aligned}
\Pi_{zz} =& e_{5,z}^2 \left[f_5^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_5^{neq}(\mathbf{r},t) \right] + e_{6,z}^2 \left[f_6^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_6^{neq}(\mathbf{r},t) \right] \\
&+ e_{7,z}^2 \left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_7^{neq}(\mathbf{r},t) \right] + e_{8,z}^2 \left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_8^{neq}(\mathbf{r},t) \right] \\
&+ e_{9,z}^2 \left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_9^{neq}(\mathbf{r},t) \right] + e_{10,z}^2 \left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{10}^{neq}(\mathbf{r},t) \right] \\
&+ e_{11,z}^2 \left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{11}^{neq}(\mathbf{r},t) \right] + e_{12,z}^2 \left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{12}^{neq}(\mathbf{r},t) \right] \\
&+ e_{13,z}^2 \left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{13}^{neq}(\mathbf{r},t) \right] + e_{14,z}^2 \left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{14}^{neq}(\mathbf{r},t) \right] \\
\\
&= (1) \left[f_1^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_1^{neq}(\mathbf{r},t) \right] + (1) \left[f_3^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_3^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_7^{neq}(\mathbf{r},t) \right] + (1) \left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_8^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_9^{neq}(\mathbf{r},t) \right] + (1) \left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{10}^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{11}^{neq}(\mathbf{r},t) \right] + (1) \left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{12}^{neq}(\mathbf{r},t) \right] \\
&+ (1) \left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{13}^{neq}(\mathbf{r},t) \right] + (1) \left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right) f_{14}^{neq}(\mathbf{r},t) \right]
\end{aligned}
\tag{A.7}
$$

$$
\begin{aligned}
&\Pi_{xy} = \Pi_{yx} = \\
&e_{7,x}e_{7,y}\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] + e_{8,x}e_{8,y}\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+e_{7,x}e_{7,y}\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] + e_{8,x}e_{8,y}\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+e_{9,x}e_{9,y}\left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_9^{neq}(\mathbf{r},t)\right] + e_{10,x}e_{10,y}\left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{10}^{neq}(\mathbf{r},t)\right] \\
&+e_{11,x}e_{11,y}\left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{11}^{neq}(\mathbf{r},t)\right] + e_{12,x}e_{12,y}\left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{12}^{neq}(\mathbf{r},t)\right] \\
&+e_{13,x}e_{13,y}\left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{13}^{neq}(\mathbf{r},t)\right] + e_{14,x}e_{14,y}\left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{14}^{neq}(\mathbf{r},t)\right]
\end{aligned}
$$

$$
\begin{aligned}
&= (1)\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_9^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{10}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{11}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{12}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{13}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{14}^{neq}(\mathbf{r},t)\right]
\end{aligned}
\tag{A.8}
$$

$$
\begin{aligned}
&\Pi_{xz} = \Pi_{zx} = \\
&e_{7,x}e_{7,z}\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] + e_{8,x}e_{8,z}\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+e_{7,x}e_{7,z}\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] + e_{8,x}e_{8,z}\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+e_{9,x}e_{9,z}\left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_9^{neq}(\mathbf{r},t)\right] + e_{10,x}e_{10,z}\left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{10}^{neq}(\mathbf{r},t)\right] \\
&+e_{11,x}e_{11,z}\left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{11}^{neq}(\mathbf{r},t)\right] + e_{12,x}e_{12,z}\left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{12}^{neq}(\mathbf{r},t)\right] \\
&+e_{13,x}e_{13,z}\left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{13}^{neq}(\mathbf{r},t)\right] + e_{14,x}e_{14,z}\left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{14}^{neq}(\mathbf{r},t)\right]
\end{aligned}
$$

$$
\begin{aligned}
&= (1)\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_9^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{10}^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{11}^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{12}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{13}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{14}^{neq}(\mathbf{r},t)\right]
\end{aligned}
\tag{A.9}
$$

$$
\begin{aligned}
&\Pi_{yz} = \Pi_{zy} = \\
&e_{7,y}e_{7,z}\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] + e_{8,y}e_{8,z}\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+e_{7,y}e_{7,z}\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] + e_{8,y}e_{8,z}\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+e_{9,y}e_{9,z}\left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_9^{neq}(\mathbf{r},t)\right] + e_{10,y}e_{10,z}\left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{10}^{neq}(\mathbf{r},t)\right] \\
&+e_{11,y}e_{11,z}\left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{11}^{neq}(\mathbf{r},t)\right] + e_{12,y}e_{12,z}\left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{12}^{neq}(\mathbf{r},t)\right] \\
&+e_{13,y}e_{13,z}\left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{13}^{neq}(\mathbf{r},t)\right] + e_{14,x}e_{14,z}\left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{14}^{neq}(\mathbf{r},t)\right]
\end{aligned}
$$

$$
\begin{aligned}
&= (1)\left[f_7^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_7^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_8^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_8^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_9^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_9^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{10}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{10}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{11}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{11}^{neq}(\mathbf{r},t)\right] \\
&+ (-1)\left[f_{12}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{12}^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_{13}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{13}^{neq}(\mathbf{r},t)\right] \\
&+ (1)\left[f_{14}^{eq}(\mathbf{r},t) + \left(1-\frac{1}{2\tau}\right)f_{14}^{neq}(\mathbf{r},t)\right]
\end{aligned}
\tag{A.10}
$$

A.3 Non-Dimensional form of the N-S Equations

Using the Chapman-Enskog expansion [129], it can be shown that the LBE corresponds directly with the N-S equations. Using the LBFS approach, we are interested in the fluxes generated by the LBE. The fluxes will be influenced by the non-dimensionalisation procedure. How they are influenced will be shown in this section.

A.3.1 Non-Dimensionalisation of LBE form of the N-S Equations

Starting with the N-S equations derived in finite difference form in Appendix A.5 gives:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 \tag{A.11}$$

$$\frac{\partial \rho \mathbf{V}}{\partial t} + \nabla \cdot (\rho \mathbf{V}\mathbf{V}) = -\nabla \rho + \nabla \left[\mu \left(\nabla \mathbf{V} + [\nabla \mathbf{V}]^T \right) \right] \tag{A.12}$$

where $\mu = \rho\nu = \rho c_s^2 \delta t \tau (1 - \frac{1}{2\tau})$ and $p = \rho c_s^2$. The LBM is predominantly solved in terms of "lattice units". This is where the physical variables in the simulation are non-dimensionalised by reference values in terms of the lattice units. The process involves converting the physical variables into non-dimensional form and then solving the LBE using variables which are in the form of the lattice units in the lattice velocity model chosen.

Let * indicate a non-dimensional variable. The following is the non-dimensionalisation procedure:

$$x^* = \frac{x}{L_{ref}} \qquad y^* = \frac{y}{L_{ref}} \qquad z^* = \frac{z}{L_{ref}}$$

$$u^* = \frac{u}{L_{ref}/t_{ref}} \qquad v^* = \frac{v}{L_{ref}/t_{ref}} \qquad w^* = \frac{w}{L_{ref}/t_{ref}}$$

$$\rho^* = \frac{\rho}{\rho_{ref}} \qquad t^* = \frac{t}{t_{ref}}$$

where t_{ref} is the reference time, L_{ref} is the reference length, and ρ_{ref} is the reference density. Applying the non-dimensionalisation procedure to Equation (A.11)

gives the mass conservation equation in non-dimensional form:

$$\frac{\partial \rho^* \rho_{ref}}{\partial t^* t_{ref}} + \frac{1}{L_{ref}} \nabla \cdot \left(\rho^* \rho_{ref} \mathbf{V}^* \frac{L_{ref}}{t_{ref}} \right) = 0 \tag{A.13}$$

and dividing across by $\frac{\rho_{ref}}{t_{ref}}$ gives:

$$\frac{\partial \rho^*}{\partial t^*} + \nabla \cdot (\rho^* \mathbf{V}^*) = 0 \tag{A.14}$$

Applying the non-dimensionalisation procedure to Equation (A.12) gives the momentum equation in non-dimensional form:

$$\begin{aligned} &\frac{\partial \rho^* \rho_{ref} \mathbf{V}^* \frac{L_{ref}}{t_{ref}}}{\partial t^* t_{ref}} + \frac{1}{L_{ref}} \nabla \cdot \left(\rho^* \rho_{ref} \mathbf{V}^* \frac{L_{ref}}{t_{ref}} \mathbf{V}^* \frac{L_{ref}}{t_{ref}} \right) = \\ &- \frac{1}{L_{ref}} \nabla \rho^* \rho_{ref} + \\ &\frac{1}{L_{ref}} \nabla \left[\rho^* \rho_{ref} \left(c_s^* \right)^2 \frac{L_{ref}^2}{t_{ref}^2} \delta t^* t_{ref} \tau^* (1 - \frac{1}{2\tau^*}) \left(\frac{1}{L_{ref}} \nabla \mathbf{V}^* \frac{L_{ref}}{t_{ref}} + \left[\frac{1}{L_{ref}} \nabla \mathbf{V}^* \frac{L_{ref}}{t_{ref}} \right]^T \right) \right] \end{aligned} \tag{A.15}$$

and dividing across by $\frac{\rho_{ref} L_{ref}}{t_{ref}^2}$ gives:

$$\begin{aligned} &\frac{\partial \rho^* \mathbf{V}^*}{\partial t^*} + \nabla \cdot (\rho^* \mathbf{V}^* \mathbf{V}^*) = \\ &- \nabla \rho^* \ + \nabla \left[\rho^* \left(c_s^* \right)^2 \delta t^* \tau^* (1 - \frac{1}{2\tau^*}) \left(\nabla \mathbf{V}^* + [\nabla \mathbf{V}^*]^T \right) \right] \end{aligned} \tag{A.16}$$

The outcome of the above means that the LBE formulated using non-dimensional terms is equivalent to what would be traditionally known as the dimensional form of the N-S equations. This means that there is no factoring of the viscous flux term by the Mach and Reynolds number as would happen in the traditional non-dimensional form of the N-S equations.

Converting Equation (A.14) into integral form gives:

$$\frac{d}{dt^*} \int \int_{V^*} \rho^* dV^* + \int_{S^*} \rho^* \mathbf{V}^* \, dS^* = 0 \tag{A.17}$$

while converting Equation (A.16) into integral form gives:

$$\begin{aligned}\frac{d}{dt^*}\int\int_{V^*}\rho^*\mathbf{V}^*dV^* + \int_{S^*}\left[\rho^*\mathbf{V}^*\mathbf{V}^* + \rho^*\left(c_s^*\right)^2\right]\ dS^* \\ - \int_{S^*}\left[\rho^*\left(c_s^*\right)^2\delta t^*\tau^*\left(1-\frac{1}{2\tau^*}\right)\left(\nabla\mathbf{V}^* + \left[\nabla\mathbf{V}^*\right]^T\right)\right]\ dS^* \\ = 0\end{aligned} \tag{A.18}$$

or rearranging into inviscid and viscous flux contributions:

$$\frac{d}{dt^*}\int\int_{V^*}(\mathbf{Q}^*dV^*) + \int_{S^*}\mathbf{F}_I^*\cdot\mathbf{n}\ dS^* + \int_{S^*}\mathbf{F}_V^*\cdot\mathbf{n}\ dS^* = 0 \tag{A.19}$$

A.3.2 Traditional Non-Dimensionalisation of the N-S equations

The approach in Appendix A.3.1 is in contrast to the traditional approach to the non-dimensionalisation of the N-S equations. This approach is as follows; let * indicate a non-dimensional variable. The following is the non-dimensionalisation procedure:

$$\begin{aligned} x^* &= \frac{x}{L_{ref}} & y^* &= \frac{y}{L_{ref}} & z^* &= \frac{z}{L_{ref}} \\ u^* &= \frac{u}{c_{s,ref}} & v^* &= \frac{v}{c_{s,ref}} & w^* &= \frac{w}{c_{s,ref}} \\ \rho^* &= \frac{\rho}{\rho_{ref}} & t^* &= \frac{tc_{s,ref}}{L_{ref}} & \mu^* &= \frac{\mu}{\mu_{ref}} \end{aligned}$$

where $c_{s,ref}$ is the speed of sound of the fluid in the free stream L_{ref} is the reference length, ρ_{ref} is the reference density, and μ_{ref} is the reference dynamic viscosity. Following the same steps as for the LBE in Appendix A.3.1 and applying to Equation (2.1) gives the N-S equations in non-dimensional form:

$$\frac{d}{dt^*}\int\int_{V^*}(\mathbf{Q}^*dV^*) + \int_{S^*}\mathbf{F}_I^*\cdot\mathbf{n}\ dS^* + \frac{Ma}{Re}\int_{S^*}\mathbf{F}_V^*\cdot\mathbf{n}\ dS^* = 0 \tag{A.20}$$

where $Ma = \dfrac{|\mathbf{V}_{max}|}{c_{s,ref}}$, $Re = \dfrac{\rho_{ref}|\mathbf{V}_{max}|L_{ref}}{\mu_{ref}}$ and $|\mathbf{V}_{max}|$ is the characteristic velocity of the fluid with respect to problem domain.

A.3.3 Reasons for Difference in Non-Dimensionalisation Procedures

Two separate approaches to non-dimensionalisation have been discussed and have significant consequences for the LBFS approach. If we compare the two non-dimensionalised equations, we see that there is a difference in how the contribution of the viscous flux is calculated:

$$LBE : \frac{d}{dt^*}\int\int_{V^*}(\mathbf{Q}^* dV^*) + \int_{S^*}\mathbf{F}_I^* \cdot \mathbf{n}\; dS^* + \int_{S^*}\mathbf{F}_V^* \cdot \mathbf{n}\; dS^* = 0 \qquad \text{(A.21)}$$

$$N-S : \frac{d}{dt^*}\int\int_{V^*}(\mathbf{Q}^* dV^*) + \int_{S^*}\mathbf{F}_I^* \cdot \mathbf{n}\; dS^* + \frac{Ma}{Re}\int_{S^*}\mathbf{F}_V^* \cdot \mathbf{n}\; dS^* = 0 \quad \text{(A.22)}$$

There are two reasons for this difference. The first reason is the fact that the LBE approach has a viscosity that can be calculated in terms of ρ^*, $\left(c_s^*\right)^2$, δt^* and τ^*. In comparison, a reference viscosity is used in the traditional non-dimensional approach. This reference viscosity is why the Ma/Re coefficient appears in the viscous flux term.

The second reason is that the LBE approach uses a reference length and time to non-dimensionalise the velocities whereas the traditional approach uses a reference speed of sound. So why can't a reference speed of sound be used with the LBE non-dimensionalisation approach? This is due to what is called compressibility error. This error is generated by the lack of third order term in the equilibrium distribution function (see Equation (2.7)). This error is of the order $O(V^3)$. This in comparison to spatial discretisation error and time discretisation error which have errors of $O(\delta x^2)$ and $O(\delta t^2)$ respectively.

Before commenting on this further the concepts of acoustic and diffusive scaling will be introduced. Acoustic scaling means that δt scales $\propto \delta x$ where-as diffusive scaling means that δt scales $\propto \delta x^2$. What this means is that if one uses acoustic scaling (N-S approach) and increase the mesh density, the compressibility error will remain constant. If diffusive scaling (LBE approach) is used and mesh density is increased, the compressibility error will reduce by the same order as the spatial discretisation error. It is desirable for all errors to be of the same order or else the compressibility error will drown out the solution and cause instability in the simulation.

A.3.4 Commentary on Applicability of Diffusive scaling

The immediate question that now comes to mind is does diffusive modelling accurately capture the physical reality that is trying to be modelled? As described by Kruger [186],

> If one is modelling incompressible flow, the speed of sound does not have any physical significance and any compressibility effects are undesired. Diffusive scaling is then the method of choice to reduce compressibility effects proportional to δx^2. If, however, the speed of sound is a physically relevant parameter (as in compressible fluid dynamics and acoustics), acoustic scaling must be chosen as it maintains the correct scaling of the speed of sound. Holdych et al.[174] emphasised that the numerical solution can only converge to the solution of the incompressible N-S Equations when δt scales $\propto \delta x^{\Psi}$ with $\Psi > 1$ since the compressibility error remains constant for $\Psi \leq 1$.

As the end goal is to model incompressible blood flow, it is recommended to use diffusive scaling. As long as the Reynolds number of the lattice simulation matches the physical Reynolds number, the law of similarity dictates that they are equivalent.

An alternative approach to removing the compressibility error was developed by He and Luo [148]. They proposed an alternative equilibrium distribution function that solved steady-state incompressible flow with zero compressibility error. However, their approach does not remove compressibility error for transient problems and diffusive scaling is required.

A.3.5 Derivation of Non-Dimensional Relaxation Factor

A very important parameter in the LBE is the relaxation factor τ. This has a big impact on the contribution of the viscous flux term and is closely related to viscosity as shown in non-dimensional form below:

$$\nu^* = \rho^* \left(c_s^*\right)^2 \delta t^* \tau^* (1 - \frac{1}{2\tau^*}) \tag{A.23}$$

This can then be manipulated into the following form:

$$\tau^* = 0.5 + \frac{\nu^*}{(c_s^*)^2 \, \delta t} \tag{A.24}$$

Rearranging gives:

$$\tau^* = 0.5 + \frac{\mathbf{V}^* D^*}{Re^* \, (c_s^*)^2 \, \delta t} \tag{A.25}$$

where D^* is the characteristic linear dimension of the fluid flow. A representation of D^* in terms of nodes and cell volumes is given in Figure A.1. For a uniform

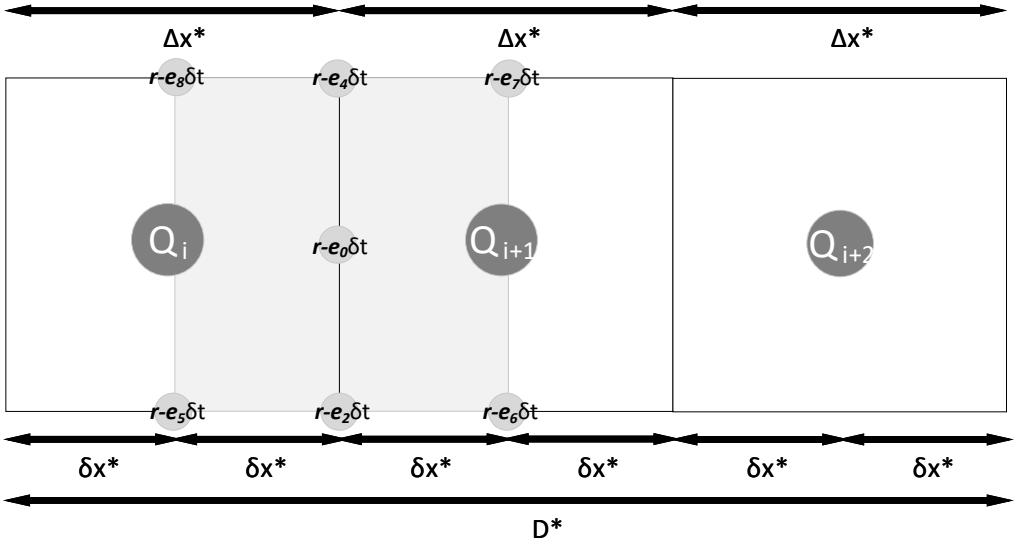


FIGURE A.1: Characteristic linear dimension and relation to node and cell volume size

grid where $\delta x^* = (\Delta x/2)$; $D^* = (N_{nodes} - 1)\delta x^* = (N_{nodes} - 1)(1) = 2N_{cells}$ and N_{nodes} = number of nodes along the characteristic linear dimension and N_{cells} = number of cell volumes along the characteristic linear dimension length. This gives:

$$\tau^* = 0.5 + \frac{\mathbf{V}^* 2N_{cells}}{Re^* \, (c_s^*)^2 \, (1)} \tag{A.26}$$

which is equivalent to:

$$\tau^* = 0.5 + \frac{Ma^* 2N_{cells}}{Re^* \, (c_s^*)} \tag{A.27}$$

which is also equivalent to:

$$\tau^* = 0.5 + \frac{\sqrt{3} Ma^* 2N_{cells}}{Re^*} \tag{A.28}$$

The Reynolds number and the Reynolds number calculated with dimensional values must have the same values. But as mentioned in Appendix A.3.3, the Mach number calculated with non-dimensional values will differ from the dimensional Mach number calculated with dimensional values for incompressible flow due to the requirement of diffusive scaling.

A.4 Hybrid Green-Gauss/Weighted-Least-Squares Gradient Operator

The Hybrid Green-Gauss/Weighted-Least-Squares Gradient Operator is given as:

$$\left[\beta_i \mathbf{M}_i + 2\left(1-\beta_i\right) V_i \mathbf{I}\right] \nabla \mathbf{Q}_i = \sum_{j=1}^{N_{faces}} \left(\beta_i \alpha_{wj} \mathbf{x}_{ij} + 2\left(1-\beta_i\right) \alpha_{Gj} \mathbf{n}_j\right) S_j \Delta \mathbf{Q}_j \quad \text{(A.29)}$$

where $\mathbf{I}$ is the identity matrix, i is the index of the current cell where the gradient is being calculated, j is the face index of cell i, $\nabla \mathbf{Q}_i$ is the gradient vector of the macroscopic variables at the centroid of cell i, S_j is the surface area of face j, α_{Gj} is a constant assumed to be equal to 0.5 for all faces, $\mathbf{n}_j$ is the normal to face j, $\Delta \mathbf{Q}_j$ is the change in macroscopic variable between the centroid of the current cell i and the centroid of the neighbouring cell with shared face j, β_i is the switching parameter for cell i and is given by:

$$\beta_i = min\left(1, \frac{2}{AR}\right) \quad \text{(A.30)}$$

where AR is the effective aspect ratio of the cell and is given by:

$$AR = \frac{2 \cdot max|\mathbf{r}_j - \mathbf{r}_i| \cdot max\left(S_j\right)}{V_i} \quad \text{(A.31)}$$

$\mathbf{r}_j$ is a position vector for the centroid of the neighbouring cell with shared face j, $\mathbf{r}_i$ is a position vector for the centroid of the current cell i and V_i is the cell volume of the current cell i. $\mathbf{M}_i$ is a matrix that weights the contribution of each neighbouring cell to the least-squares gradient calculation and is given by:

$$\mathbf{M}_i = \begin{bmatrix} \sum_{j=1}^{N_{faces}} \omega_j \Delta x \Delta x & \sum_{j=1}^{N_{faces}} \omega_j \Delta y \Delta x & \sum_{j=1}^{N_{faces}} \omega_j \Delta z \Delta x \\ \sum_{j=1}^{N_{faces}} \omega_j \Delta x \Delta y & \sum_{j=1}^{N_{faces}} \omega_j \Delta y \Delta y & \sum_{j=1}^{N_{faces}} \omega_j \Delta z \Delta y \\ \sum_{j=1}^{N_{faces}} \omega_j \Delta x \Delta z & \sum_{j=1}^{N_{faces}} \omega_j \Delta y \Delta z & \sum_{j=1}^{N_{faces}} \omega_j \Delta z \Delta z \end{bmatrix} \quad \text{(A.32)}$$

Δx, Δy and Δz are the changes in location along the Cartesian axes between the current cell's centroid and the j^{th} neighbouring cell's centroid. α_{wj} is an interpolation factor given by:

$$\alpha_{wj} = 4\left(\frac{|\mathbf{r}_k - \mathbf{r}_i| \cdot \mathbf{n}_j}{|\mathbf{r}_j - \mathbf{r}_i| \cdot \mathbf{n}_j}\right)^2 \quad \text{(A.33)}$$

where $\mathbf{r}_k$ is the centroid of the face j and ω_j is a weighting function given by:

$$\omega_j = \alpha_{wj} \frac{S_j}{|\mathbf{r}_j - \mathbf{r}_i|} \tag{A.34}$$

Finally $\mathbf{x}_{ij}$ is given by:

$$\mathbf{x}_{ij} = \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|} \tag{A.35}$$

A.5 Non-Conservative form of N-S Equations

This section shows how the coefficients of the spatial derivatives of the primitive variables are calculated for the non-conservative form of the N-S equations. This was then used in Section 2.5.1 to calculate the RK4 stability criteria. It is not possible to calculate the correct stability using the coefficients in their base form. Therefore a symmetric form of the coefficients is used in Section 2.5 to analyse the stability of the N-S equations. As detailed in Abarbanel [178], the non-conservative form of the N-S equations are given as follows:

$$\begin{aligned}\frac{\partial \mathbf{Q}}{\partial t} + A\frac{\partial \mathbf{Q}}{\partial x} + B\frac{\partial \mathbf{Q}}{\partial y} + J\frac{\partial \mathbf{Q}}{\partial z} = \\ + C\frac{\partial^2 \mathbf{Q}}{\partial^2 x} + D\frac{\partial^2 \mathbf{Q}}{\partial^2 y} + K\frac{\partial^2 \mathbf{Q}}{\partial^2 z} + E_{xy}\frac{\partial^2 \mathbf{Q}}{\partial x \partial y} + E_{yz}\frac{\partial^2 \mathbf{Q}}{\partial y \partial z} + E_{zx}\frac{\partial^2 \mathbf{Q}}{\partial z \partial x}\end{aligned} \tag{A.36}$$

where:

$$\mathbf{Q} = \begin{bmatrix} \rho \\ u \\ v \\ w \end{bmatrix} \tag{A.37}$$

$$A = \begin{bmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix} \tag{A.38}$$

$$B = \begin{bmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & v \end{bmatrix} \tag{A.39}$$

$$J = \begin{bmatrix} w & 0 & 0 & \rho \\ 0 & w & 0 & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & w \end{bmatrix} \tag{A.40}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{2\mu}{\rho} & 0 & 0 \\ 0 & 0 & \frac{\mu}{\rho} & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \end{bmatrix} \tag{A.41}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \\ 0 & 0 & \frac{2\mu}{\rho} & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \end{bmatrix} \tag{A.42}$$

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \\ 0 & 0 & \frac{\mu}{\rho} & 0 \\ 0 & 0 & 0 & \frac{2\mu}{\rho} \end{bmatrix} \tag{A.43}$$

$$E_{xy} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\mu}{\rho} & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.44}$$

$$E_{yz} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \\ 0 & 0 & \frac{\mu}{\rho} & 0 \end{bmatrix} \tag{A.45}$$

$$E_{zx} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \\ 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \end{bmatrix} \tag{A.46}$$

A symmetric operator is defined as follows:

$$S_p = \begin{bmatrix} \frac{\rho}{c_s} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.47}$$

$$S_p^{-1} = \begin{bmatrix} \frac{c_s}{\rho} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.48}$$

Applying this symmetric operator simultaneously to the matrices A through E gives:

$$S_p^{-1} A S_p = \begin{bmatrix} u & c_s & 0 & 0 \\ c_s & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix} \tag{A.49}$$

$$S_p^{-1} B S_p = \begin{bmatrix} v & 0 & c_s & 0 \\ 0 & v & 0 & 0 \\ c_s & 0 & v & 0 \\ 0 & 0 & 0 & v \end{bmatrix} \tag{A.50}$$

$$S_p^{-1} J S_p = \begin{bmatrix} w & 0 & 0 & c_s \\ 0 & w & 0 & 0 \\ 0 & 0 & w & 0 \\ c_s & 0 & 0 & w \end{bmatrix} \tag{A.51}$$

$$S_p^{-1} C S_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{2\mu}{\rho} & 0 & 0 \\ 0 & 0 & \frac{\mu}{\rho} & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \end{bmatrix} \tag{A.52}$$

$$S_p^{-1} D S_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \\ 0 & 0 & \frac{2\mu}{\rho} & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \end{bmatrix} \tag{A.53}$$

$$S_p^{-1} K S_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \\ 0 & 0 & \frac{\mu}{\rho} & 0 \\ 0 & 0 & 0 & \frac{2\mu}{\rho} \end{bmatrix} \tag{A.54}$$

$$S_p^{-1} E_{xy} S_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\mu}{\rho} & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.55}$$

$$S_p^{-1} E_{yz} S_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \\ 0 & 0 & \frac{\mu}{\rho} & 0 \end{bmatrix} \tag{A.56}$$

$$S_p^{-1} E_{zx} S_p = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\mu}{\rho} \\ 0 & 0 & 0 & 0 \\ 0 & \frac{\mu}{\rho} & 0 & 0 \end{bmatrix} \tag{A.57}$$

A.6 VBA RK4 Stability Region Code

The following code, is written in VBA, and was used to calculate the stability contour for a RK4 integration method. The graphical representation is given by Figure 2.2.

```
Sub rk4()
Dim wb As Workbook
Set wb = ThisWorkbook
```

```
Dim ws As Worksheet

Set ws = wb.Sheets("Runge Kutta 4")

Dim j As Long

j = 1

Dim x, y As Double

Dim g_re, g_im, g_mod As Double

For x = -300 To 300 Step 1

For y = -300 To 300 Step 1

a = x / 100

b = y / 100

g_re = 1 + b

g_re = g_re + 0.5 * (b ∧ 2 - a ∧ 2)

g_re = g_re + (b ∧ 3 - 3 * a ∧ 2 * b) / 6

g_re = g_re + (a ∧ 4 + b ∧ 4 - 6 * a ∧ 2 * b ∧ 2) / 24

g_im = a + a * b + (3 * b ∧ 2 * a - a ∧ 3) / 6 + (4 * b ∧ 3 * a - 4 *
a ∧ 3 * b) / 24

g_mod = (g_re ∧ 2 + g_im ∧ 2) ∧ 0.5

If Abs(g_mod - 1) < 0.01 Then

ws.Cells(j, 1).Value = g_re

ws.Cells(j, 2).Value = g_im

ws.Cells(j, 3).Value = x

ws.Cells(j, 4).Value = y

ws.Cells(j, 5).Value = g_mod

j = j + 1

End If

Next

Next

End Sub
```

Appendix B

Source Code Repository

B.1 Source Code

The code used in this work is kept in a repository under MIT license at https://github.com/CHRG-Developer. A view of the architecture of the program is provided in Figure B.1. Note that the following dependencies are required to be installed before use:

- Boost -https://www.boost.org/users/download/
- CUDA -https://developer.nvidia.com/cuda-downloads
- TECIO - Tecplot Input/Output library https://github.com/su2code/SU2/tree/master/externals/tecio

A makefile is provided that will compile the code. Note however that references to external library need to be updated in the makefile for local user settings.

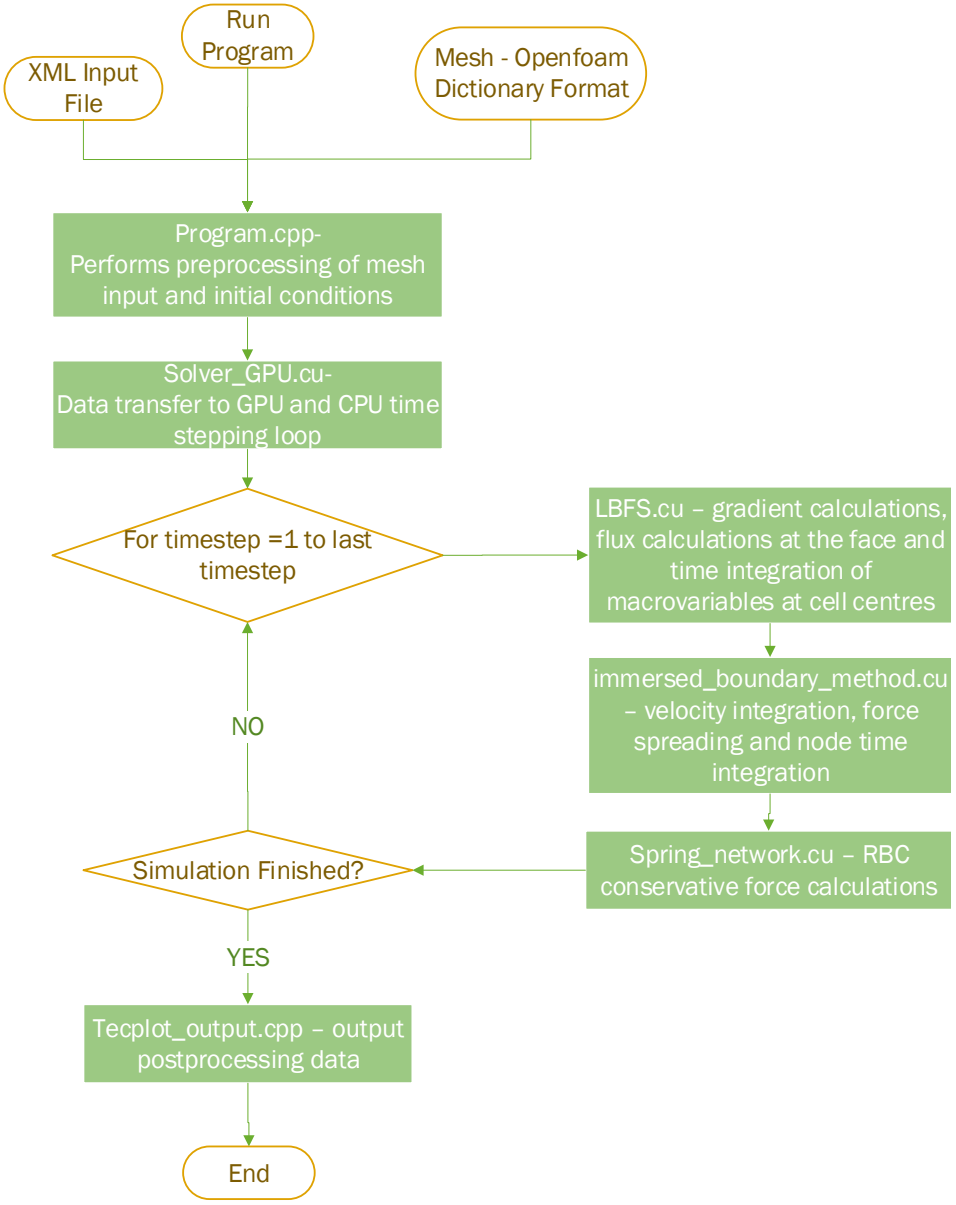


FIGURE B.1: Architecture of the LBFS ADE-SP blood flow model source code.

Bibliography

[1] Mingzhu Chen. Numerical Modelling of Red Blood Cell Structural Mechanics. *Doctoral*, jul 2018. doi: https://doi.org/10.21427/9ecm-9j76. URL https://arrow.dit.ie/engdoc/105.

[2] Yonggoo Kim, Joonhong Park, and Myungshin Kim. Diagnostic approaches for inherited hemolytic anemia in the genetic era. *Blood Research*, 52(2): 84, 2017. ISSN 2287-979X. doi: 10.5045/br.2017.52.2.84. URL https://synapse.koreamed.org/DOIx.php?id=10.5045/br.2017.52.2.84.

[3] Karl Rupp. 42 Years of Microprocessor Trend Data — Karl Rupp, 2018. URL https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/.

[4] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982. ISSN 10902716. doi: 10.1016/0021-9991(82)90058-4.

[5] Loudon Catherine Tordesillas and Antoinette. Loudon1546.pdf, 1998. ISSN 00225193.

[6] J P Mills, L Qie, M Dao, C T Lim, and S Suresh. Nonlinear elastic and viscoelastic deformation of the human red blood cell with optical tweezers. *Mechanics & chemistry of biosystems : MCB*, 1(3):169–80, sep 2004. ISSN 1546-2048. URL http://www.ncbi.nlm.nih.gov/pubmed/16783930.

[7] Weijuan Yao, Zongyao Wen, Zongyi Yan, Dagong Sun, Weibo Ka, Lide Xie, and Shu Chien. Low viscosity Ektacytometry and its validation tested by flow chamber. *Journal of Biomechanics*, 34(11):1501–1509, nov 2001. ISSN 0021-9290. doi: 10.1016/S0021-9290(01)00109-9. URL https://www.sciencedirect.com/science/article/pii/S0021929001001099.

[8] Igor V. Pivkin and George Em Karniadakis. Accurate coarse-grained modeling of red blood cells. *Physical Review Letters*, 101(11):118105, sep 2008. ISSN 00319007. doi: 10.1103/PhysRevLett.101.118105.

[9] Daniel A. Reasor, Jonathan R. Clausen, and Cyrus K. Aidun. Coupling the lattice-Boltzmann and spectrin-link methods for the direct numerical simulation of cellular blood flow. *International Journal for Numerical Methods in Fluids*, 68(6):767–781, feb 2012. ISSN 02712091. doi: 10.1002/fld.2534. URL http://doi.wiley.com/10.1002/fld.2534.

[10] Manouk Abkarian, Magalie Faivre, and Annie Viallat. Swinging of red blood cells under shear flow. *Physical Review Letters*, 98(18):188302, apr 2007. ISSN 10797114. doi: 10.1103/PhysRevLett.98.188302.

[11] Nvidia. Nvidia Tesla V100 GPU Whitepaper. Technical report, 2017.

[12] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark Computations of Laminar Flow Around a Cylinder. pages 547–566, 1996. doi: 10.1007/978-3-322-89849-4_39.

[13] Hana Thomas, Jaime Diamond, Adrianna Vieco, Shaoli Chaudhuri, Eliezer Shinnar, Sara Cromer, Pablo Perel, George A. Mensah, Jagat Narula, Catherine O. Johnson, Gregory A. Roth, and Andrew E. Moran. Global Atlas of Cardiovascular Disease 2000-2016: The Path to Prevention and Control. *Global Heart*, 13(3):143–163, 2018. ISSN 22118179. doi: 10.1016/j.gheart.2018.09.511.

[14] James F Antaki, Michael R Ricci, Josiah E Verkaik, Shaun T Snyder, M Timothy, Jeongho Kim, Dave B Paden, Marina V Kameneva, Bradley E Paden, D Peter, and Harvey S Borovetz. *NIH Public Access*, volume 1. 2011. ISBN 1323901000. doi: 10.1007/s13239-010-0011-9.PediaFlow.

[15] Eoin A. Murphy, Adrian S. Dunne, David M. Martin, and Fergal J. Boyle. Oxygen Mass Transport in Stented Coronary Arteries. *Annals of Biomedical Engineering*, 44(2):508–522, 2016. ISSN 15739686. doi: 10.1007/s10439-015-1501-6.

[16] A. M. Gambaruto and A. J. João. Flow structures in cerebral aneurysms. *Computers and Fluids*, 2012. ISSN 00457930. doi: 10.1016/j.compfluid.2012.02.020.

[17] Hernán G. Morales and Odile Bonnefous. Unraveling the relationship between arterial flow and intra-aneurysmal hemodynamics. *Journal of Biomechanics*, 2015. ISSN 18732380. doi: 10.1016/j.jbiomech.2015.01.016.

[18] Jianping Xiang, Sabareesh K. Natarajan, Markus Tremmel, Ding Ma, J. Mocco, L. Nelson Hopkins, Adnan H. Siddiqui, Elad I. Levy, and Hui Meng. Hemodynamic-morphologic discriminants for intracranial aneurysm rupture. *Stroke*, 2011. ISSN 00392499. doi: 10.1161/STROKEAHA.110.592923.

[19] Nick Ashton. Physiology of red and white blood cells. *Anaesthesia and Intensive Care Medicine*, 14(6):261–266, 2013. ISSN 14720299. doi: 10.1016/j.mpaic.2013.03.001. URL http://dx.doi.org/10.1016/j.mpaic.2013.03.001.

[20] A. V. Belyaev, J. L. Dunster, J. M. Gibbins, M. A. Panteleev, and V. Volpert. Modeling thrombosis in silico: Frontiers, challenges, unresolved problems and milestones. *Physics of Life Reviews*, 2018. ISSN 15710645. doi: 10.1016/j.plrev.2018.02.005.

[21] Oguz K. Baskurt and Herbert J. Meiselman. Blood Rheology and Hemodynamics. *Seminars in Thrombosis and Hemostasis*, 29(5):435–450, 2003. ISSN 00946176. doi: 10.1055/s-2003-44551.

[22] G. D.O. Lowe. 1 Blood rheology in vitro and in vivo. *Bailliere's Clinical Haematology*, 1(3):597–636, 1987. ISSN 09503536. doi: 10.1016/S0950-3536(87)80018-5.

[23] E. W. Errill. Rheology of blood. *Physiological Reviews*, 49(4):863–888, oct 1969. ISSN 0031-9333. doi: 10.1152/physrev.1969.49.4.863. URL http://www.ncbi.nlm.nih.gov/pubmed/4898603http://www.physiology.org/doi/10.1152/physrev.1969.49.4.863.

[24] Peter W. Rand, Eleanor Lacombe, Hamilton E. Hunt, and William H. Austin. Viscosity of normal human blood under normothermic and hypothermic conditions. *Journal of Applied Physiology*, 19(1):117–122, jan 1964. ISSN 8750-7587. doi: 10.1152/jappl.1964.19.1.117. URL http://www.physiology.org/doi/10.1152/jappl.1964.19.1.117.

[25] S. Chien, S. Usami, H. M. Taylor, J. L. Lundberg, and M. I. Gregersen. Effects of hematocrit and plasma proteins on human blood rheology at low shear rates. *Journal of applied physiology*, 21(1):81–87, 1966. ISSN 00218987.

[26] H. Schmid-Schönbein, R. Wells, and J. Goldstone. Influence of deformability of human red cells upon blood viscosity. *Circulation research*, 25(2):131–143, 1969. ISSN 00097330. doi: 10.1161/01.RES.25.2.131.

[27] Aleksander S Popel and Paul C Johnson. Microcirculation and Hemorheology. *Annual review of fluid mechanics*, 37:43–69, jan 2005. ISSN 0066-4189. doi: 10.1146/annurev.fluid.37.042604.133933. URL http://www.ncbi.nlm.nih.gov/pubmed/21151769http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3000688.

[28] Stuart R. Keller and Richard Skalak. Motion of a tank-treading ellipsoidal particle in a shear flow. *Journal of Fluid Mechanics*, 120:27–47, jul 1982. ISSN 0022-1120. doi: 10.1017/S0022112082002651. URL https://www.cambridge.org/core/product/identifier/S0022112082002651/type/journal{_}article.

[29] Richard E. Waugh and Robert M. Hochmuth. Mechanics and deformability of hematocytes. In *Biomechanics: Principles and Practices*. 2014. ISBN 9781439870990. doi: 10.1201/b15575.

[30] S. J. Singer and Garth L. Nicolson. The fluid mosaic model of the structure of cell membranes. *Science*, 1972. ISSN 00368075. doi: 10.1126/science.175.4023.720.

[31] Christopher W. Harland, Miranda J. Bradley, and Raghuveer Parthasarathy. Phospholipid bilayers are viscoelastic. *Proceedings of the National Academy of Sciences of the United States of America*, 2010. ISSN 10916490. doi: 10.1073/pnas.1010700107.

[32] E A Evans, R Waugh, and L Melnik. Elastic area compressibility modulus of red cell membrane. *Biophysical journal*, 16(6):585–95, jun 1976. ISSN 0006-3495. doi: 10.1016/S0006-3495(76)85713-X. URL http://www.ncbi.nlm.nih.gov/pubmed/1276386http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1334882.

[33] Volkmar Heinrich, Ken Ritchie, Narla Mohandas, and Evan Evans. Elastic Thickness Compressibilty of the Red Cell Membrane. *Biophysical Journal*,

81(3):1452–1463, sep 2001. ISSN 00063495. doi: 10.1016/S0006-3495(01)75800-6. URL http://www.ncbi.nlm.nih.gov/pubmed/11509359.

[34] David Boal. *Mechanics of the Cell*. Cambridge University Press, Cambridge, 2012. ISBN 9781139022217. doi: 10.1017/CBO9781139022217. URL http://ebooks.cambridge.org/ref/id/CBO9781139022217.

[35] R. Skalak, A. Tozeren, R. P. Zarda, and S. Chien. Strain Energy Function of Red Blood Cell Membranes. *Biophysical Journal*, 1973. ISSN 00063495. doi: 10.1016/S0006-3495(73)85983-1.

[36] X. Z. Li, D. Barthes-Biesel, and A. Helmy. Large deformations and burst of a capsule freely suspended in an elongational flow. *Journal of Fluid Mechanics*, 1988. ISSN 14697645. doi: 10.1017/S0022112088000394.

[37] Christophe Quéguiner and Dominique Barthès-Biesel. Axisymmetric motion of capsules through cylindrical channels. *Journal of Fluid Mechanics*, 1997. ISSN 00221120. doi: 10.1017/S0022112097006587.

[38] J. F. Gross and N. Ozkaya. Flow of axisymmetric red blood cells in narrow capillaries. *Journal of Fluid Mechanics*, 1986. ISSN 14697645. doi: 10.1017/S0022112086002355.

[39] P. B. Canham. The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell. *Journal of Theoretical Biology*, 1970. ISSN 10958541. doi: 10.1016/S0022-5193(70)80032-7.

[40] W. Helfrich. Elastic Properties of Lipid Bilayers: Theory and Possible Experiments. *Zeitschrift fur Naturforschung - Section C Journal of Biosciences*, 1973. ISSN 18657125. doi: 10.1515/znc-1973-11-1209.

[41] S. Svetina and B. Zeks. Bilayer couple hypothesis of red cel shape transformations and osmotic hemolysis. *Biomedica Biochimica Acta*, 1983. ISSN 0232766X.

[42] S. Svetina and B. Žekš. Membrane bending energy and shape determination of phospholipid vesicles and red blood cells. *European Biophysics Journal*, 1989. ISSN 01757571. doi: 10.1007/BF00257107.

[43] H. W.Gerald Lim, Michael Wortis, and Ranjan Mukhopadhyay. Stomatocyte-discocyte-echinocyte sequence of the human red blood cell: Evidence for the bilayer-couple hypothesis from membrane mechanics. *Proceedings of the National Academy of Sciences of the United States of America*, 2002. ISSN 00278424. doi: 10.1073/pnas.202617299.

[44] Ling Miao, Udo Seifert, Michael Wortis, and Hans Günther Döbereiner. Budding transitions of fluid-bilayer vesicles: The effect of area-difference elasticity. *Physical Review E*, 1994. ISSN 1063651X. doi: 10.1103/PhysRevE.49.5389.

[45] Udo Seifert, Karin Berndl, and Reinhard Lipowsky. Shape transformations of vesicles: Phase diagram for spontaneous- curvature and bilayer-coupling models. *Physical Review A*, 1991. ISSN 10502947. doi: 10.1103/PhysRevA.44.1182.

[46] C. Pozrikidis. Numerical simulation of the flow-induced deformation of red blood cells. *Annals of Biomedical Engineering*, 2003. ISSN 00906964. doi: 10.1114/1.1617985.

[47] Martin Kraus, Wolfgang Wintz, Udo Seifert, and Reinhard Lipowsky. Fluid vesicles in shear flow. *Physical Review Letters*, 1996. ISSN 10797114. doi: 10.1103/PhysRevLett.77.3685.

[48] S. Ramanujan and C. Pozrikidis. Deformation of liquid capsules enclosed by elastic membranes in simple shear flow: Large deformations and the effect of fluid viscosities. *Journal of Fluid Mechanics*, 1998. ISSN 00221120. doi: 10.1017/S0022112098008714.

[49] Junfeng Zhang, Paul C. Johnson, and Aleksander S. Popel. Effects of erythrocyte deformability and aggregation on the cell free layer and apparent viscosity of microscopic blood flows. *Microvascular Research*, 2009. ISSN 00262862. doi: 10.1016/j.mvr.2009.01.010.

[50] Prosenjit Bagchi and R. Murthy Kalluri. Dynamics of nonspherical capsules in shear flow. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 2009. ISSN 15393755. doi: 10.1103/PhysRevE.80.016307.

[51] Johannes Mauer, Simon Mendez, Luca Lanotte, Franck Nicoud, Manouk Abkarian, Gerhard Gompper, and Dmitry A. Fedosov. Flow-Induced Transitions of Red Blood Cell Shapes under Shear. *Physical Review Letters*, 121

(11):118103, sep 2018. ISSN 0031-9007. doi: 10.1103/PhysRevLett.121.118103. URL http://www.ncbi.nlm.nih.gov/pubmed/30265089https://link.aps.org/doi/10.1103/PhysRevLett.121.118103.

[52] C. Pozrikidis. Numerical simulation of cell motion in tube flow. *Annals of Biomedical Engineering*, 2005. ISSN 00906964. doi: 10.1007/s10439-005-8975-6.

[53] C. Pozrikidis. Axisymmetric motion of a file of red blood cells through capillaries. *Physics of Fluids*, 2005. ISSN 10706631. doi: 10.1063/1.1830484.

[54] Junfeng Zhang, Paul C. Johnson, and Aleksander S. Popel. Red blood cell aggregation and dissociation in shear flows simulated by lattice Boltzmann method. *Journal of Biomechanics*, 2008. ISSN 00219290. doi: 10.1016/j.jbiomech.2007.07.020.

[55] Pierre Taraconat, Jean-Philippe Gineys, Damien Isebe, Franck Nicoud, and Simon Mendez. Numerical simulation of deformable particles in a Coulter counter. *International Journal for Numerical Methods in Biomedical Engineering*, aug 2019. ISSN 2040-7939. doi: 10.1002/cnm.3243. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.3243.

[56] Meongkeun Ju, Swe Soe Ye, Bumseok Namgung, Seungkwan Cho, Hong Tong Low, Hwa Liang Leo, and Sangho Kim. A review of numerical methods for red blood cell flow simulation. *Computer Methods in Biomechanics and Biomedical Engineering*, 2015. ISSN 14768259. doi: 10.1080/10255842.2013.783574.

[57] Yohsuke Imai, Toshihiro Omori, Yuji Shimogonya, Takami Yamaguchi, and Takuji Ishikawa. Numerical methods for simulating blood flow at macro, micro, and multi scales. *Journal of Biomechanics*, 2016. ISSN 18732380. doi: 10.1016/j.jbiomech.2015.11.047.

[58] P. Dimitrakopoulos. Interfacial dynamics in Stokes flow via a three-dimensional fully-implicit interfacial spectral boundary element algorithm. *Journal of Computational Physics*, 2007. ISSN 10902716. doi: 10.1016/j.jcp.2006.12.004.

[59] Jonathan B. Freund. Numerical Simulation of Flowing Blood Cells. *Annual Review of Fluid Mechanics*, 2014. ISSN 0066-4189. doi: 10.1146/annurev-fluid-010313-141349.

[60] Shravan K. Veerapaneni, Abtin Rahimian, George Biros, and Denis Zorin. A fast algorithm for simulating vesicle flows in three dimensions. *Journal of Computational Physics*, 2011. ISSN 10902716. doi: 10.1016/j.jcp.2011.03.045.

[61] Dmitry A. Fedosov, Bruce Caswell, Aleksander S. Popel, and George E M Karniadakis. Blood Flow and Cell-Free Layer in Microvessels. *Microcirculation*, 2010. ISSN 10739688. doi: 10.1111/j.1549-8719.2010.00056.x.

[62] Xuejin Li, Petia M. Vlahovska, and George Em Karniadakis. Continuum- and particle-based modeling of shapes and dynamics of red blood cells in health and disease, 2013. ISSN 17446848.

[63] Dmitry A. Fedosov, Bruce Caswell, and George Em Karniadakis. A multiscale red blood cell model with accurate mechanics, rheology,dynamics. *Biophysical Journal*, 2010. ISSN 15420086. doi: 10.1016/j.bpj.2010.02.002.

[64] Dennis E. Discher, David H. Boal, and Seng K. Boey. Simulations of the erythrocyte cytoskeleton at large deformation. II. Micropipette aspiration. *Biophysical Journal*, 1998. ISSN 00063495. doi: 10.1016/S0006-3495(98)74076-7.

[65] Mingzhu Chen and Fergal Boyle. An enhanced spring-particle model for red blood cell structrual mechanics. *Journal of Biomechanical Engineering*, 139(December), 2017. ISSN 0148-0731. doi: 10.1115/1.4037590. URL http://biomechanical.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4037590.

[66] M. Dao, J. Li, and S. Suresh. Molecularly based analysis of deformation of spectrin network and human erythrocyte. *Materials Science and Engineering C*, 2006. ISSN 09284931. doi: 10.1016/j.msec.2005.08.020.

[67] J. Li, M. Dao, C. T. Lim, and Subra Suresh. Spectrin-level modeling of the cytoskeleton and optical tweezers stretching of the erythrocyte. *Biophysical Journal*, 2005. ISSN 00063495. doi: 10.1529/biophysj.104.047332.

[68] Witold Dzwinel, Krzysztof Boryczko, and David A. Yuen. A discrete-particle model of blood dynamics in capillary vessels. *Journal of Colloid and Interface Science*, 2003. ISSN 00219797. doi: 10.1016/S0021-9797(02)00075-9.

[69] Hiroshi Noguchi and Gerhard Gompper. Shape transitions of fluid vesicles and red blood cells in capillary flows. *Proceedings of the National Academy of Sciences of the United States of America*, 2005. ISSN 00278424. doi: 10.1073/pnas.0504243102.

[70] Ken Ichi Tsubota, Shigeo Wada, and Takami Yamaguchi. Simulation study on effects of hematocrit on blood flow using particle method. In *Proceedings of the 2005 Summer Bioengineering Conference*, 2005. ISBN 0974249211.

[71] Michael M. Dupin, Ian Halliday, Chris M. Care, Lyuba Alboul, and Lance L. Munn. Modeling the flow of dense suspensions of deformable particles in three dimensions. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 2007. ISSN 15393755. doi: 10.1103/PhysRevE.75.066707.

[72] Igor V. Pivkin and George Em Karniadakis. Accurate coarse-grained modeling of red blood cells. *Physical Review Letters*, 2008. ISSN 00319007. doi: 10.1103/PhysRevLett.101.118105.

[73] Dmitry a Fedosov, Hiroshi Noguchi, and Gerhard Gompper. Multiscale modeling of blood flow: from single cells to blood rheology. *Biomechanics and modeling in mechanobiology*, 2013. ISSN 1617-7940. doi: 10.1007/s10237-013-0497-9. URL http://www.ncbi.nlm.nih.gov/pubmed/23670555.

[74] Dmitry A. Fedosov, Matti Peltomäki, and Gerhard Gompper. Deformation and dynamics of red blood cells in flow through cylindrical microchannels. *Soft Matter*, 2014. ISSN 17446848. doi: 10.1039/c4sm00248b.

[75] Dmitry A. Fedosov, Wenxiao Pan, Bruce Caswell, Gerhard Gompper, and George E. Karniadakis. Predicting human blood viscosity in silico. *Proceedings of the National Academy of Sciences of the United States of America*, 2011. ISSN 00278424. doi: 10.1073/pnas.1101210108.

[76] D. A. Fedosov, B. Caswell, S. Suresh, and G. E. Karniadakis. Quantifying the biophysical characteristics of Plasmodium-falciparum- parasitized red blood cells in microcirculation. *Proceedings of the National Academy of Sciences of the United States of America*, 2011. ISSN 00278424. doi: 10.1073/pnas.1009492108.

[77] Dmitry A. Fedosov, Huan Lei, Bruce Caswell, Subra Suresh, and George E. Karniadakis. Multiscale modeling of red blood cell mechanics and blood

flow in malaria. *PLoS Computational Biology*, 2011. ISSN 1553734X. doi: 10.1371/journal.pcbi.1002270.

[78] Dmitry A. Fedosov, Bruce Caswell, and George Em Karniadakis. Wall shear stress-based model for adhesive dynamics of red blood cells in malaria. *Biophysical Journal*, 2011. ISSN 15420086. doi: 10.1016/j.bpj.2011.03.027.

[79] C. (Constantine) Pozrikidis. *Boundary integral and singularity methods for linearized viscous flow*. Cambridge University Press, 1992. ISBN 0521406935. URL https://tinyurl.com/yc8lkybn.

[80] C. Pozrikidis. Numerical Simulation of the Flow-Induced Deformation of Red Blood Cells. *Annals of Biomedical Engineering*, 31(10):1194–1205, nov 2003. ISSN 0090-6964. doi: 10.1114/1.1617985. URL http://link.springer.com/10.1114/1.1617985.

[81] W R Dodson, P Dimitrakopoulos, and P. Dimitrakopoulos. Tank-treading of erythrocytes in strong shear flows via a nonstiff cytoskeleton-based continuum computational modeling. *Biophysical journal*, 99(9): 2906–16, nov 2010. ISSN 1542-0086. doi: 10.1016/j.bpj.2010.08.048. URL http://www.ncbi.nlm.nih.gov/pubmed/21044588http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2966011.

[82] T. Omori, T. Ishikawa, D. Barthès-Biesel, A.-V. Salsac, Y. Imai, and T. Yamaguchi. Tension of red blood cell membrane in simple shear flow. *Physical Review E*, 86(5):056321, nov 2012. ISSN 1539-3755. doi: 10.1103/PhysRevE.86.056321. URL https://link.aps.org/doi/10.1103/PhysRevE.86.056321.

[83] Daan Frenkel and Berend Smit. *Understanding molecular simulation : from algorithms to applications*. ISBN 0080519989. URL https://tinyurl.com/ybw4nqyv.

[84] Ting Ye, Nhan Phan-Thien, and Chwee Teck Lim. Particle-based simulations of red blood cells—A review. *Journal of Biomechanics*, 49(11):2255–2266, jul 2016. ISSN 00219290. doi: 10.1016/j.jbiomech.2015.11.050. URL https://linkinghub.elsevier.com/retrieve/pii/S0021929015006922.

[85] P. J. Hoogerbrugge and J. M.V.A. Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics, 1992. ISSN 12864854.

[86] A. Malevanets and J. M. Yeomans. Dynamics of short polymer chains in solution. jul 2000. doi: 10.1209/epl/i2000-00428-0. URL http://arxiv.org/abs/cond-mat/0007123http://dx.doi.org/10.1209/epl/i2000-00428-0.

[87] Hiroshi Noguchi and Gerhard Gompper. Dynamics of fluid vesicles in shear flow: Effect of membrane viscosity and thermal fluctuations. *Physical Review E*, 72(1):011901, jul 2005. ISSN 1539-3755. doi: 10.1103/PhysRevE.72.011901. URL https://link.aps.org/doi/10.1103/PhysRevE.72.011901.

[88] Matti Peltomäki and Gerhard Gompper. Sedimentation of single red blood cells. *Soft Matter*, 9(34):8346, aug 2013. ISSN 1744-683X. doi: 10.1039/c3sm50592h. URL http://xlink.rsc.org/?DOI=c3sm50592h.

[89] S. Majid Hosseini and James J. Feng. A particle-based model for the transport of erythrocytes in capillaries. *Chemical Engineering Science*, 2009. ISSN 00092509. doi: 10.1016/j.ces.2008.11.028.

[90] S. Majid Hosseini and James J. Feng. How Malaria Parasites Reduce the Deformability of Infected Red Blood Cells. *Biophysical Journal*, 103(1):1–10, jul 2012. ISSN 0006-3495. doi: 10.1016/J.BPJ.2012.05.026. URL https://www.sciencedirect.com/science/article/pii/S0006349512005796.

[91] Tenghu Wu and James J. Feng. Simulation of malaria-infected red blood cells in microfluidic channels: Passage and blockage. *Biomicrofluidics*, 7(4):044115, jul 2013. ISSN 1932-1058. doi: 10.1063/1.4817959. URL http://aip.scitation.org/doi/10.1063/1.4817959.

[92] Davod ALIZADEHRAD, Yohsuke IMAI, Keita NAKAAKI, Takuji ISHIKAWA, and Takami YAMAGUCHI. Parallel Simulation of Cellular Flow in Microvessels Using a Particle Method. *Journal of Biomechanical Science and Engineering*, 7(1):57–71, 2012. ISSN 1880-9863. doi: 10.1299/jbse.7.57. URL http://joi.jlc.jst.go.jp/JST.JSTAGE/jbse/7.57?from=CrossRef.

[93] Yohsuke Imai, Hitoshi Kondo, Takuji Ishikawa, Chwee Teck Lim, and Takami Yamaguchi. Modeling of hemodynamics arising from malaria infection. *Journal of Biomechanics*, 43(7):1386–1393, may 2010. ISSN 00219290. doi: 10.1016/j.jbiomech.2010.01.011. URL https://linkinghub.elsevier.com/retrieve/pii/S002192901000045X.

[94] Yohsuke Imai, Keita Nakaaki, Hitoshi Kondo, Takuji Ishikawa, Chwee Teck Lim, and Takami Yamaguchi. Margination of red blood cells infected by Plasmodium falciparum in a microvessel. *Journal of Biomechanics*, 44(8): 1553–1558, may 2011. ISSN 0021-9290. doi: 10.1016/J.JBIOMECH.2011.02.084. URL https://www.sciencedirect.com/science/article/pii/S002192901100176X.

[95] Hiroki Kamada, Yohsuke Imai, Masanori Nakamura, Takuji Ishikawa, and Takami Yamaguchi. Computational analysis on the mechanical interaction between a thrombus and red blood cells: Possible causes of membrane damage of red blood cells at microvessels. *Medical Engineering & Physics*, 34 (10):1411–1420, dec 2012. ISSN 1350-4533. doi: 10.1016/J.MEDENGPHY.2012.01.003. URL https://www.sciencedirect.com/science/article/pii/S1350453312000069.

[96] J. Donea, S. Giuliani, and J.P. Halleux. An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1-3):689–723, sep 1982. ISSN 0045-7825. doi: 10.1016/0045-7825(82)90128-1. URL https://www.sciencedirect.com/science/article/pii/0045782582901281.

[97] J De Hart, G W M Peters, P J G Schreurs, and F P T Baaijens. A three-dimensional computational analysis of fluid-structure interaction in the aortic valve. *Journal of biomechanics*, 36(1):103–12, jan 2003. ISSN 0021-9290. doi: 10.1016/s0021-9290(02)00244-0. URL http://www.ncbi.nlm.nih.gov/pubmed/12485644.

[98] Cyrus K. Aidun and Dewei W. Qi. A new method for analysis of the fluid interaction with a deformable membrane. *Journal of Statistical Physics*, 1998. ISSN 00224715.

[99] Daniel A. Reasor, Jonathan R. Clausen, and Cyrus K. Aidun. Coupling the lattice-Boltzmann and spectrin-link methods for the direct numerical simulation of cellular blood flow. *International Journal for Numerical Methods in Fluids*, 2012. ISSN 02712091. doi: 10.1002/fld.2534.

[100] Robert Miles MacMeccan. Mechanistic Effects of Erythrocytes on Platelet Deposition in Coronary Thrombosis Mechanistic Effects of Erythrocytes on Platelet Deposition in Coronary Thrombosis. 2007.

[101] Charles S. Peskin. The immersed boundary method. In Arieh Iserles, editor, *Acta Numerica 2002*, pages 479–518. Cambridge University Press, Cambridge, 2010. doi: 10.1017/cbo9780511550140.007. URL https://www.cambridge.org/core/product/identifier/CBO9780511550140A011/type/book{_}part.

[102] Swe Soe Ye, Meongkeun Ju, and Sangho Kim. Recovery of cell-free layer and wall shear stress profile symmetry downstream of an arteriolar bifurcation. *Microvascular Research*, 106:14–23, jul 2016. ISSN 00262862. doi: 10.1016/j.mvr.2016.03.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0026286216300188.

[103] Gábor Závodszky, Britt van Rooij, Ben Czaja, Victor Azizi, David de Kanter, and Alfons G. Hoekstra. Red blood cell and platelet diffusivity and margination in the presence of cross-stream gradients in blood flows. *Physics of Fluids*, 31(3):031903, mar 2019. ISSN 1070-6631. doi: 10.1063/1.5085881. URL http://aip.scitation.org/doi/10.1063/1.5085881.

[104] M. Navidbakhsh and M. Rezazadeh. An immersed boundary-lattice Boltzmann model for simulation of malaria-infected red blood cell in microchannel. *Scientia Iranica*, 19(5):1329–1336, oct 2012. ISSN 1026-3098. doi: 10.1016/J.SCIENT.2012.08.001. URL https://www.sciencedirect.com/science/article/pii/S1026309812001757.

[105] Lucy Zhang, Axel Gerstenberger, Xiaodong Wang, and Wing Kam Liu. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, 2004. ISSN 00457825. doi: 10.1016/j.cma.2003.12.044.

[106] Yaling Liu and Wing Kam Liu. Rheology of red blood cell aggregation by computer simulation. *Journal of Computational Physics*, 2006. ISSN 10902716. doi: 10.1016/j.jcp.2006.05.010.

[107] R. Glowinski, T.-W. Pan, T.I. Hesla, and D.D. Joseph. A distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755–794, aug 1999. ISSN 0301-9322. doi: 10.1016/S0301-9322(98)00048-2. URL https://www.sciencedirect.com/science/article/pii/S0301932298000482?via{%}3Dihub.

[108] Xing Shi, Guang Lin, Jianfeng Zou, and Dmitry A. Fedosov. A lattice Boltzmann fictitious domain method for modeling red blood cell deformation

and multiple-cell hydrodynamic interactions in flow. *International Journal for Numerical Methods in Fluids*, 72(8):895–911, jul 2013. ISSN 02712091. doi: 10.1002/fld.3764. URL http://doi.wiley.com/10.1002/fld.3764.

[109] Katharine H. Fraser, M. Ertan Taskin, Bartley P. Griffith, and Zhongjun J. Wu. The use of computational fluid dynamics in the development of ventricular assist devices. *Medical Engineering & Physics*, 33(3):263–280, apr 2011. ISSN 1350-4533. doi: 10.1016/J.MEDENGPHY.2010.10.014. URL https://www.sciencedirect.com/science/article/pii/S1350453310002456.

[110] J. A. Backer, C. P. Lowe, H. C. J. Hoefsloot, and P. D. Iedema. Combined length scales in dissipative particle dynamics. *The Journal of Chemical Physics*, 123(11):114905, sep 2005. ISSN 0021-9606. doi: 10.1063/1.2013208. URL http://aip.scitation.org/doi/10.1063/1.2013208.

[111] Zhen Li, Xin Bian, Yu-Hang Tang, and George Em Karniadakis. A dissipative particle dynamics method for arbitrarily complex geometries. *Journal of Computational Physics*, 355:534–547, feb 2018. ISSN 0021-9991. doi: 10.1016/J.JCP.2017.11.014. URL https://www.sciencedirect.com/science/article/pii/S0021999117308525.

[112] S. Mendez, E. Gibaud, and F. Nicoud. An unstructured solver for simulations of deformable particles in flows at arbitrary Reynolds numbers. *Journal of Computational Physics*, 2014. ISSN 10902716. doi: 10.1016/j.jcp.2013.08.061.

[113] Charles Hirsch. *Numerical Computation of Internal and External Flows*, volume 1. 2007. ISBN 978-3-540-85055-7. doi: 10.1007/978-3-540-85056-4. URL http://link.springer.com/10.1007/978-3-540-85056-4.

[114] Suhas Patankar. Numerical heat transfer and fluid flow, 1980. ISSN 0009-286X.

[115] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*, volume 113. 2016. ISBN 978-3-319-16873-9. doi: 10.1007/978-3-319-16874-6. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84939129919{&}partnerID=tZOtx3y1.

[116] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics: Seventh Edition*. 2013. ISBN 9781856176354. doi: 10.1016/C2009-0-26328-8.

[117] A. J. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems, 1967. ISSN 00219991.

[118] D Kwak and C C Kiris. Computation of Viscous Incompressible Flows. 2010. URL https://books.google.co.uk/books?id=eG4AeiZt9m4C.

[119] R. C. Swanson and E. Turkel. Multistage Schemes with Multigrid for Euler and Navier-Strokes Equations: Components and Analysis. (NASA Technical Paper 3631), 1997.

[120] Karel Kozel, Petr Louda, Jarom, and Ír P R Íhoda. NUMERICAL SOLUTION OF AN UNSTEADY FLOW USING ARTIFICIAL COMPRESSIBILITY METHOD. URL https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.657.

[121] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 1972. ISSN 00179310. doi: 10.1016/0017-9310(72)90054-3.

[122] J. P. Van Doormaal and G. D. Raithby. Enhancements of the simple method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 1984. ISSN 01495720. doi: 10.1080/01495728408961817.

[123] C. R. Maliska and G. D. Raithby. A method for computing three dimensional flows using non-orthogonal boundary-fitted co-ordinates. *International Journal for Numerical Methods in Fluids*, 1984. ISSN 10970363. doi: 10.1002/fld.1650040606.

[124] R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 1986. ISSN 10902716. doi: 10.1016/0021-9991(86)90099-9.

[125] Dochan Kwak, Cetin Kiris, and Chang Sung Kim. Computational challenges of viscous incompressible flows. *Computers & Fluids*, 34(3):283–299, mar 2005. ISSN 0045-7930. doi: 10.1016/J.COMPFLUID.2004.05.008. URL https://www.sciencedirect.com/science/article/pii/S0045793004000817.

[126] Z. Guo and C. Shu. *Lattice Boltzmann method and its applications in engineering*, volume 54. 2013. ISBN 1001-6538. doi: 10.1007/s11434-009-0681-6.

[127] Y. H. Qian, D. D'Humières, and P. Lallemand. Lattice bgk models for navier-stokes equation. *EPL*, 1992. ISSN 12864854. doi: 10.1209/0295-5075/17/6/001.

[128] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. *The Lattice Boltzmann Method*. Graduate Texts in Physics. Springer International Publishing, Cham, 2017. ISBN 978-3-319-44647-9. doi: 10.1007/978-3-319-44649-3. URL http://link.springer.com/10.1007/978-3-319-44649-3.

[129] L. Jehring. Chapman, S.; Cowling, T. G., The Mathematical Theory of Non-Uniform Gases. 3rd edition. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 72(11):610–610, 1992. ISSN 00442267. doi: 10.1002/zamm.19920721124. URL http://doi.wiley.com/10.1002/zamm.19920721124{%}5Cnhttp://scitation.aip.org/content/aapt/journal/ajp/30/5/10.1119/1.1942035.

[130] Gongwen Peng, Haowen Xi, Comer Duncan, and So-Hsiang Chou. Finite volume scheme for the lattice Boltzmann method on unstructured meshes. *Physical Review E*, 59(4):4675–4682, 1999. ISSN 1063-651X. doi: 10.1103/PhysRevE.59.4675. URL http://link.aps.org/doi/10.1103/PhysRevE.59.4675.

[131] Gongwen Peng, Haowen Xi, Comer Duncan, and So-Hsiang Chou. Lattice Boltzmann method on irregular meshes. *Physical Review E*, 58(4):R4124–R4127, oct 1998. ISSN 1063-651X. doi: 10.1103/PhysRevE.58.R4124. URL https://link.aps.org/doi/10.1103/PhysRevE.58.R4124.

[132] Maik Stiebler, Jonas Tölke, and Manfred Krafczyk. An upwind discretization scheme for the finite volume lattice Boltzmann method. *Computers and Fluids*, 2006. ISSN 00457930. doi: 10.1016/j.compfluid.2005.09.002.

[133] Dhiraj V. Patil and K. N. Lakshmisha. Finite volume TVD formulation of lattice Boltzmann simulation on unstructured mesh. *Journal of Computational Physics*, 228(14):5262–5279, 2009. ISSN 00219991. doi: 10.1016/j.jcp.2009.04.008.

[134] S. Ubertini, G. Bella, and S. Succi. Unstructured lattice Boltzmann equation with memory. *Mathematics and Computers in Simulation*, 72(2-6):237–241, 2006. ISSN 03784754. doi: 10.1016/j.matcom.2006.05.009.

[135] A. Zarghami, M. J. Maghrebi, J. Ghasemi, and S. Ubertini. Lattice Boltzmann finite volume formulation with improved stability. *Communications in Computational Physics*, 12(1):42–64, 2012. ISSN 18152406. doi: 10.4208/cicp.151210.140711a.

[136] Yan Wang, Liming Yang, and Chang Shu. From lattice Boltzmann method to lattice Boltzmann flux solver. *Entropy*, 17(11):7713–7735, 2015. ISSN 10994300. doi: 10.3390/e17117713.

[137] C. Shu, Y. Wang, C. J. Teo, and J. Wu. Development of lattice boltzmann flux solver for simulation of incompressible flows. *Advances in Applied Mathematics and Mechanics*, 6(4):436–460, 2014. ISSN 20751354. doi: 10.4208/aamm.2014.4.s2.

[138] Nicolas Pellerin, Sébastien Leclaire, and Marcelo Reggio. Solving incompressible fluid flows on unstructured meshes with the lattice Boltzmann flux solver. *Engineering Applications of Computational Fluid Mechanics*, 11(1): 310–327, 2017. ISSN 1994-2060. doi: 10.1080/19942060.2017.1292410.

[139] Qi-Feng Wu, Chang Shu, Yan Wang, and Li-Ming Yang. An effective lattice Boltzmann flux solver on arbitrarily unstructured meshes. *Modern Physics Letters B*, 32(12n13):1840012, may 2018. ISSN 0217-9849. doi: 10.1142/S0217984918400122.

[140] Yan Wang, Chang Shu, Chiang Juay Teo, Jie Wu, and Liming Yang. Three-Dimensional Lattice Boltzmann Flux Solver and Its Applications to Incompressible Isothermal and Thermal Flows. *Communications in Computational Physics*, 18(3):593–620, 2015. ISSN 19917120. doi: 10.4208/cicp.300514.160115a.

[141] Eli Turkel. Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of Computational Physics*, 72(2):277–298, oct 1987. ISSN 10902716. doi: 10.1016/0021-9991(87)90084-2.

[142] A. G. Malan, R. W. Lewis, and P. Nithiarasu. An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part I. Theory and implementation. *International Journal*

for Numerical Methods in Engineering, 54(5):695–714, jun 2002. ISSN 0029-5981. doi: 10.1002/nme.447. URL http://doi.wiley.com/10.1002/nme.447.

[143] K. Hejranfar and K. Parseh. Application of a preconditioned high-order accurate artificial compressibility-based incompressible flow solver in wide range of Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 86(1):46–77, jan 2018. ISSN 02712091. doi: 10.1002/fld.4407. URL http://doi.wiley.com/10.1002/fld.4407.

[144] Xiaoyi He, Gary D. Doolen, and T. Clark. Comparison of the lattice Boltzmann method and the artificial compressibility method for Navier-Stokes equations. *Journal of Computational Physics*, 2002. ISSN 00219991. doi: 10.1006/jcph.2002.7064.

[145] Abdullah Shah, Li Yuan, and Aftab Khan. Upwind compact finite difference scheme for time-accurate solution of the incompressible Navier-Stokes equations. *Applied Mathematics and Computation*, 215(9):3201–3213, jan 2010. ISSN 00963003. doi: 10.1016/j.amc.2009.10.001.

[146] Kazem Hejranfar and Kaveh Parseh. Preconditioned characteristic boundary conditions based on artificial compressibility method for solution of incompressible flows. *Journal of Computational Physics*, 2017. ISSN 10902716. doi: 10.1016/j.jcp.2017.05.014.

[147] Harriet E. Judy and Noreen B. Price. HEMOGLOBIN LEVEL AND RED BLOOD CELL COUNT FINDINGS IN NORMAL WOMEN. *Journal of the American Medical Association*, 167(5):563, may 1958. ISSN 0002-9955. doi: 10.1001/jama.1958.02990220033010. URL http://jama.jamanetwork.com/article.aspx?doi=10.1001/jama.1958.02990220033010.

[148] Xiaoyi He and Li-Shi Luo. Lattice Boltzmann Model for the Incompressible Navier–Stokes Equation. *Journal of Statistical Physics*, 88(3/4):927–944, 1997. ISSN 0022-4715. doi: 10.1023/B:JOSS.0000015179.12689.e4. URL http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:JOSS.0000015179.12689.e4.

[149] Eli Turkel. Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of Computational Physics*, 72(2):277–298, 1987. ISSN 10902716. doi: 10.1016/0021-9991(87)90084-2.

[150] M Bernaschi, S Succi, and H Chen. Accelerated lattice Boltzmann schemes for steady-state flow simulations. *Journal of Scientific Computing*, 16(2): 135–144, 2001. ISSN 08857474. doi: 10.1023/A:1012230722915.

[151] R Verberg and a J Ladd. Simulation of low-Reynolds-number flow via a time-independent lattice-Boltzmann method. *Physical Review E*, 60(3): 3366–3373, 1999. ISSN 1063-651X. doi: 10.1103/PhysRevE.60.3366.

[152] DAVID R. Noble and DAVID J. Holdych. Full Newton lattice Boltzmann method for time-steady flows using a direct linear solver. *International Journal of Modern Physics C*, 18(04):652–660, apr 2007. ISSN 0129-1831. doi: 10.1142/S0129183107010905.

[153] Jonas Tölke, Manfred Krafczyk, and Ernst Rank. A multigrid-solver for the discrete Boltzmann equation. *Journal of Statistical Physics*, 107(1/2): 573–591, 2002. ISSN 00224715. doi: 10.1023/A:1014551813787.

[154] Taehun Lee and Ching-Long Lin. An eulerian description of the streaming process in the lattice Boltzmann equation. *Journal of Computational Physics*, 185(2):445–471, mar 2003. ISSN 0021-9991. doi: 10.1016/S0021-9991(02)00065-7.

[155] Takeshi Seta and Ryoichi Takahashi. Numerical Stability Analysis of FDLBM. *Journal of Statistical Physics*, 107(1/2):557–572, 2002. ISSN 00224715. doi: 10.1023/A:1014599729717.

[156] Jonas Tölke, Manfred Krafczyk, Manuel Schulz, Ernst Rank, and Rodolfo Berrios. Implicit discretization and nonuniform mesh refinement approaches for FD discretizations of LBGK Models. *International Journal of Modern Physics C*, 09(08):1143–1157, dec 1998. ISSN 0129-1831. doi: 10.1142/S0129183198001059.

[157] Dimitri J. Mavriplis. Multigrid solution of the steady-state lattice Boltzmann equation. *Computers and Fluids*, 35(8-9):793–804, 2006. ISSN 00457930. doi: 10.1016/j.compfluid.2005.07.020.

[158] Zhaoli Guo, T. S. Zhao, and Yong Shi. Preconditioned lattice-Boltzmann method for steady flows. *Physical Review E*, 70(6):1–8, 2004. ISSN 15393755. doi: 10.1103/PhysRevE.70.066706.

[159] Kannan N. Premnath, Martin J. Pattison, and Sanjoy Banerjee. Steady state convergence acceleration of the generalized lattice Boltzmann equation with forcing term through preconditioning. *Journal of Computational Physics*, 228(3):746–769, 2009. ISSN 00219991. doi: 10.1016/j.jcp.2008.09.028.

[160] Salvador Izquierdo and Norberto Fueyo. Preconditioned Navier-Stokes schemes from the generalised lattice Boltzmann equation. *Progress in Computational Fluid Dynamics, An International Journal*, 8(1-4):189, 2008. ISSN 1468-4349. doi: 10.1504/PCFD.2008.018089.

[161] Salvador Izquierdo and Norberto Fueyo. Optimal preconditioning of lattice Boltzmann methods. *Journal of Computational Physics*, 228(17):6479–6495, 2009. ISSN 00219991. doi: 10.1016/j.jcp.2009.05.040.

[162] Alessandro De Rosis. Preconditioned lattice Boltzmann method for steady flows: A noncascaded central-moments-based approach. *Physical Review E*, 96(6):063308, dec 2017. ISSN 2470-0045. doi: 10.1103/PhysRevE.96.063308.

[163] Farzaneh Hajabdollahi and Kannan N. Premnath. Improving the low Mach number steady state convergence of the cascaded lattice Boltzmann method by preconditioning. *Computers & Mathematics with Applications*, feb 2017. ISSN 0898-1221. doi: 10.1016/J.CAMWA.2016.12.034.

[164] Farzaneh Hajabdollahi and Kannan N. Premnath. Galilean-invariant preconditioned central-moment lattice Boltzmann method without cubic velocity errors for efficient steady flow simulations. *Physical Review E*, 97(5), 2018. ISSN 24700053. doi: 10.1103/PhysRevE.97.053303.

[165] Xuhui Meng, Liang Wang, Xiaofan Yang, and Zhaoli Guo. Preconditioned multiple-relaxation-time lattice Boltzmann equation model for incompressible flow in porous media. *Physical Review E*, 98(5):1–12, 2018. ISSN 24700053. doi: 10.1103/PhysRevE.98.053309.

[166] Erik P. DeBenedictis and Erik P. It's Time to Redefine Moore's Law Again. *Computer*, 50(2):72–75, feb 2017. ISSN 0018-9162. doi: 10.1109/MC.2017.34. URL http://ieeexplore.ieee.org/document/7842840/.

[167] K Rupp and J Weinbub. A computational scientist's perspective on current and future hardware architectures. In *Austrian HPC Meeting*, page 24, 2016. URL http://www.iue.tuwien.ac.at/pdf/ib{_}2016/hashed{_}links/5d4PC0r3w{_}{_}rCY{_}us.pdf.

[168] S. Ali Mirsoleimani, Aske Plaat, Jos Vermaseren, and Jaap Van Den Herik. Performance analysis of a 240 thread tournament level MCTS Go program on the Intel Xeon Phi. In *Modelling and Simulation 2014 - European Simulation and Modelling Conference, ESM 2014*, 2014. ISBN 9789077381861.

[169] George Teodoro, Tahsin Kurc, Jun Kong, Lee Cooper, and Joel Saltz. Comparative Performance Analysis of Intel Xeon Phi, GPU, and CPU: A Case Study from Microscopy Image Analysis. *IEEE transactions on parallel and distributed systems : a publication of the IEEE Computer Society*, 2014:1063–1072, may 2014. ISSN 1045-9219. doi: 10.1109/IPDPS.2014.111. URL http://www.ncbi.nlm.nih.gov/pubmed/25419088http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4240026.

[170] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Physical Review*, 94(3):511–525, may 1954. ISSN 0031-899X. doi: 10.1103/PhysRev.94.511. URL https://link.aps.org/doi/10.1103/PhysRev.94.511.

[171] Randall J. LeVeque. Boundary Conditions and Ghost Cells. *Finite Volume Methods for Hyperbolic Problems*, pages 129–138, 2010. doi: 10.1017/cbo9780511791253.008.

[172] Eiji Shima, Keiichi Kitamura, and Takanori Haga. Green–Gauss/Weighted-Least-Squares Hybrid Gradient Reconstruction for Arbitrary Polyhedra Unstructured Grids. *AIAA Journal*, 51(11):2740–2747, nov 2013. ISSN 0001-1452. doi: 10.2514/1.J052095. URL http://arc.aiaa.org/doi/10.2514/1.J052095.

[173] L D Landau and E M Lifshitz. *Fluid Mechanics*, volume 6. 1987. ISBN 0080339336. doi: 10.1007/b138775. URL http://www.springerlink.com/index/10.1007/b138775.

[174] David J. Holdych, David R. Noble, John G. Georgiadis, and Richard O. Buckius. Truncation error analysis of lattice Boltzmann methods. *Journal of Computational Physics*, 193(2):595–619, 2004. ISSN 00219991. doi: 10.1016/j.jcp.2003.08.012.

[175] A Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping

schemes. *Fluid Dynamics and Co-located Conferences*, M(July 2016):—-, 1981. doi: 10.2514/6.1981-1259.

[176] MIT OpenCourseware. Chapter 10: Runge Kutta methods. URL http://web.mit.edu/course/16/16.90/OldFiles/BackUp/www/pdfs/Chapter10.pdf.

[177] Johan Sowa. Stability of a runge-kutta method navier-stokes equation. 30 (December 1989):542–560, 1990.

[178] Saul Abarbanel and David Gottlieb. Optimal time splitting for two- and three-dimensional navier-stokes equations with mixed derivatives. *Journal of Computational Physics*, 41(1):1–33, 1981. ISSN 10902716. doi: 10.1016/0021-9991(81)90077-2.

[179] D. J. MAVRIPLIS and A. JAMESON. Multigrid solution of the Navier-Stokes equations on triangular meshes. *AIAA Journal*, 28(8):1415–1425, aug 1990. ISSN 0001-1452. doi: 10.2514/3.25233. URL http://arc.aiaa.org/doi/10.2514/3.25233.

[180] J. T. Dodge, B. G. Brown, E. L. Bolson, and H. T. Dodge. Lumen diameter of normal human coronary arteries. Influence of age, sex, anatomic variation, and left ventricular hypertrophy or dilation. *Circulation*, 86(1):232–246, 1992. ISSN 0009-7322. doi: 10.1161/01.CIR.86.1.232. URL http://circ.ahajournals.org/cgi/doi/10.1161/01.CIR.86.1.232.

[181] Gerard J. Tortora and Bryan Derrickson. *Principles of Anatomy & Physiology 14th Edition*. 2014. ISBN 9781118345009.

[182] R. J. Trudnowski and R. C. Rico. Specific gravity of blood and plasma at 4 and 37 C. *Clinical Chemistry*, 20(5):615–616, 1974. ISSN 00099147.

[183] C J Freitas. Perspective - Selected Benchmarks from Commercial Cfd Codes. *Journal of Fluids Engineering-Transactions of the Asme*, 117(2):208–218, 1995. ISSN 0098-2202. doi: 10.1115/1.2817132.

[184] Ashkan Javadzadegan, Andy S C Yong, Michael Chang, Austin C C Ng, John Yiannikas, Martin K C Ng, Masud Behnia, and Leonard Kritharides. Flow recirculation zone length and shear rate are differentially affected by stenosis severity in human coronary arteries. *American journal of physiology. Heart and circulatory physiology*, 304:H559–66, 2013. ISSN 1522-1539. doi: 10.

1152/ajpheart.00428.2012. URL http://www.ncbi.nlm.nih.gov/pubmed/23241317.

[185] Shardt. Matlab Taylor Vortex Code, 2016. URL https://github.com/lbm-principles-practice/code/blob/master/chapter13/matlab/vortexdecay.m.

[186] Timm Kruger and Orest Shardt. *The lattice boltzmann method*. Springer, 2016. ISBN 9783319446479. doi: 10.1007/978-3-319-44649-3.

[187] X. D. Niu, C. Shu, Y. T. Chew, and T. G. Wang. Investigation of stability and hydrodynamics of different lattice Boltzmann models. *Journal of Statistical Physics*, 117(3-4):665–680, 2004. ISSN 00224715. doi: 10.1007/s10955-004-2264-x.

[188] W a R C Ku and Richard S Hirsh. A Pseudospectral Metho of the Three-Dimensional Incompressible. *Journal of Computational Physics*, 70:439–462, 1987. ISSN 00219991. doi: 10.1016/0021-9991(87)90190-2.

[189] H. Ding, C. Shu, K. S. Yeo, and D. Xu. Numerical computation of three-dimensional incompressible viscous flows in the primitive variable form by local multiquadric differential quadrature method. *Computer Methods in Applied Mechanics and Engineering*, 195(7-8):516–533, 2006. ISSN 00457825. doi: 10.1016/j.cma.2005.02.006.

[190] S. C.R. Dennis and Gau zu Chang. Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100. *Journal of Fluid Mechanics*, 1970. ISSN 14697645. doi: 10.1017/S0022112070001428.

[191] Ratnesh K. Shukla, Mahidhar Tatineni, and Xiaolin Zhong. Very high-order compact finite difference schemes on non-uniform grids for incompressible Navier-Stokes equations. *Journal of Computational Physics*, 224(2):1064–1094, 2007. ISSN 00219991. doi: 10.1016/j.jcp.2006.11.007.

[192] Ming Chih Lai and Charles S. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*, 160(2):705–719, 2000. ISSN 00219991. doi: 10.1006/jcph.2000.6483.

[193] Zhi Gang Feng and Efstathios E. Michaelides. The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems.

Journal of Computational Physics, 195(2):602–628, 2004. ISSN 00219991. doi: 10.1016/j.jcp.2003.10.013.

[194] Jungwoo Kim, Dongjoo Kim, and Haecheon Choi. An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries. *Journal of Computational Physics*, 171(1):132–150, 2001. ISSN 00219991. doi: 10.1006/jcph.2001.6778.

[195] Zhi Gang Feng and Efstathios E. Michaelides. Robust treatment of no-slip boundary condition and velocity updating for the lattice-Boltzmann simulation of particulate flows. *Computers and Fluids*, 38(2):370–381, 2009. ISSN 00457930. doi: 10.1016/j.compfluid.2008.04.013. URL http://dx.doi.org/10.1016/j.compfluid.2008.04.013.

[196] J. Wu and C. Shu. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. *Journal of Computational Physics*, 228(6):1963–1979, 2009. ISSN 00219991. doi: 10.1016/j.jcp.2008.11.019. URL http://dx.doi.org/10.1016/j.jcp.2008.11.019.

[197] Y. Wang, C. Shu, C. J. Teo, and L. M. Yang. An efficient immersed boundary-lattice Boltzmann flux solver for simulation of 3D incompressible flows with complex geometry. *Computers and Fluids*, 124:54–66, 2016. ISSN 00457930. doi: 10.1016/j.compfluid.2015.10.009. URL http://dx.doi.org/10.1016/j.compfluid.2015.10.009.

[198] Y. Wang, C. Shu, C. J. Teo, and J. Wu. An immersed boundary-lattice Boltzmann flux solver and its applications to fluid-structure interaction problems. *Journal of Fluids and Structures*, 54:440–465, 2015. ISSN 10958622. doi: 10.1016/j.jfluidstructs.2014.12.003.

[199] Shin K. Kang and Yassin A. Hassan. A comparative study of direct-forcing immersed boundary-lattice Boltzmann methods for stationary complex boundaries. *International Journal for Numerical Methods in Fluids*, 2011. ISSN 02712091. doi: 10.1002/fld.2304.

[200] Pablo Ouro, Luis Cea, Luis Ramírez, and Xesús Nogueira. An immersed boundary method for unstructured meshes in depth averaged shallow water models. *International Journal for Numerical Methods in Fluids*, 2016. ISSN 10970363. doi: 10.1002/fld.4201.

[201] A. Pinelli, I.Z. Naqavi, U. Piomelli, and J. Favier. Immersed-boundary methods for general finite-difference and finite-volume Navier–Stokes solvers. *Journal of Computational Physics*, 229(24):9073–9091, dec 2010. ISSN 0021-9991. doi: 10.1016/J.JCP.2010.08.021. URL https://www.sciencedirect.com/science/article/pii/S0021999110004687.

[202] Wing Kam Liu, Yijung Chen, R Aziz Uras, and Tang Chang '. Computer methods in applied mechanics and engineering Generalized multiple scale reproducing kernel particle methods. *ELSEYIER Comput. Methods Appl. Mech. Engrg*, 139(96):91–157, 1996. URL https://tinyurl.com/ya4t2e7u.

[203] S. Mendez, E. Gibaud, and F. Nicoud. An unstructured solver for simulations of deformable particles in flows at arbitrary Reynolds numbers. *Journal of Computational Physics*, 256:465–483, jan 2014. ISSN 10902716. doi: 10.1016/j.jcp.2013.08.061.

[204] Francisco Toja-Silva, Julien Favier, and Alfredo Pinelli. Radial basis function (RBF)-based interpolation and spreading for the immersed boundary method. *Computers and Fluids*, 2014. ISSN 00457930. doi: 10.1016/j.compfluid.2014.09.026.

[205] Juwon Jang and Changhoon Lee. An immersed boundary method for nonuniform grids. *Journal of Computational Physics*, 341:1–12, 2017. ISSN 10902716. doi: 10.1016/j.jcp.2017.04.014. URL http://dx.doi.org/10.1016/j.jcp.2017.04.014.

[206] P. Bagchi and S. Balachandar. Effect of free rotation on the motion of a solid sphere in linear shear flow at moderate Re. *Physics of Fluids*, 14(8):2719–2737, 2002. ISSN 10706631. doi: 10.1063/1.1487378.

[207] L. Schiller and A. Naumann. Über die grundlegenden berechnungen bei der schwerkraftaufbereitung. *Zeitschrift des Vereines deutscher Ingenieure*, 1933.

[208] P. G. Saffman. The lift on a small sphere in a slow shear flow. *Journal of Fluid Mechanics*, 22(2):385–400, jun 1965. ISSN 0022-1120. doi: 10.1017/S0022112065000824. URL https://www.cambridge.org/core/product/identifier/S0022112065000824/type/journal{_}article.

[209] John B. McLaughlin. Inertial migration of a small sphere in linear shear flows. *Journal of Fluid Mechanics*, 224:261–274, mar 1991.

ISSN 0022-1120. doi: 10.1017/S0022112091001751. URL https://www.cambridge.org/core/product/identifier/S0022112091001751/type/journal{_}article.

[210] R. Mei. An approximate expression for the shear lift force on a spherical particle at finite reynolds number. *International Journal of Multiphase Flow*, 18(1):145–147, jan 1992. ISSN 0301-9322. doi: 10.1016/0301-9322(92)90012-6. URL https://www.sciencedirect.com/science/article/pii/0301932292900126.

[211] RYOICHI KUROSE and SATORU KOMORI. Drag and lift forces on a rotating sphere in a linear shear flow. *Journal of Fluid Mechanics*, 384:183–206, apr 1999. ISSN 0022-1120. doi: 10.1017/S0022112099004164. URL https://www.cambridge.org/core/product/identifier/S0022112099004164/type/journal{_}article.

[212] Samer Adeeb. *Introduction to Solid Mechanics & Finite Element Analysis*. 2019. URL https://sameradeeb-new.srv.ualberta.ca/.

[213] Osayomwanbor Ehi-Egharevba. Viscoelastic behaviour of Red Blood Cells - In Progress.

[214] Zhangli Peng, Adel Mashayekh, and Qiang Zhu. Erythrocyte responses in low-shear-rate flows: Effects of non-biconcave stress-free state in the cytoskeleton. *Journal of Fluid Mechanics*, 742:96–118, mar 2014. ISSN 00221120. doi: 10.1017/jfm.2014.14.

[215] Gerhard. Gompper and Michael. Schick. *Soft matter*. Wiley-VCH, 2006. ISBN 9783527315024. URL https://www.wiley.com/en-ie/Soft+Matter,+Volume+4:+Lipid+Bilayers+and+Red+Blood+Cells-p-9783527315024.

[216] Nadeeshani Maheshika Geekiyanage, Marie Anne Balanant, Emilie Sauret, Suvash Saha, Robert Flower, Chwee Teck Lim, and YuanTong Gu. A coarse-grained red blood cell membrane model to study stomatocyte-discocyte-echinocyte morphologies. *PLOS ONE*, 14(4):e0215447, apr 2019. ISSN 1932-6203. doi: 10.1371/journal.pone.0215447. URL http://dx.plos.org/10.1371/journal.pone.0215447.

[217] Evan Evans and Yuan Cheng Fung. Improved measurements of the erythrocyte geometry. *Microvascular Research*, 4(4):335–347, oct 1972. ISSN 10959319. doi: 10.1016/0026-2862(72)90069-6.

[218] P. Dimitrakopoulos. Analysis of the variation in the determination of the shear modulus of the erythrocyte membrane: Effects of the constitutive law and membrane modeling. *Physical Review E*, 85(4): 041917, apr 2012. ISSN 1539-3755. doi: 10.1103/PhysRevE.85.041917. URL http://www.ncbi.nlm.nih.gov/pubmed/22680508http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3605755https://link.aps.org/doi/10.1103/PhysRevE.85.041917.

[219] MAUNG YE SWE SOE. *NUMERICAL STUDY OF ERYTHROCYTE MECHANICS: DEFORMATION AND AGGREGATION*. PhD thesis. URL https://scholarbank.nus.edu.sg/handle/10635/125238.

[220] Jules Dupire, Manouk Abkarian, and Annie Viallat. A simple model to understand the effect of membrane shear elasticity and stress-free shape on the motion of red blood cells in shear flow. *Soft Matter*, 11(42):8372–8382, oct 2015. ISSN 17446848. doi: 10.1039/c5sm01407g.

[221] Daniel Cordasco, Alireza Yazdani, and Prosenjit Bagchi. Comparison of erythrocyte dynamics in shear flow under different stress-free configurations. *Physics of Fluids*, 26(4):041902, apr 2014. ISSN 10897666. doi: 10.1063/1.4871300. URL http://aip.scitation.org/doi/10.1063/1.4871300.

[222] E Gibaud. Numerical simulation of red blood cells flowing in a blood analyzer. 2015.

[223] Dmitry A. Fedosov. Multiscale Modeling of Blood Flow and Soft Matter. *Brown University*, (May):292, 2010. ISSN 1532-7914. doi: 10.1080/01635580802395717.

[224] M. A. Mader, V. Vitkova, M. Abkarian, A. Viallat, and T. Podgorski. Dynamics of viscous vesicles in shear flow. *The European Physical Journal E*, 19(4):389–397, apr 2006. ISSN 1292-8941. doi: 10.1140/epje/i2005-10058-x. URL http://link.springer.com/10.1140/epje/i2005-10058-x.

[225] Victoria Vitkova, Maud-Alix Mader, Benoît Polack, Chaouqi Misbah, and Thomas Podgorski. Micro-Macro Link in Rheology of Erythrocyte and

Vesicle Suspensions. *Biophysical Journal*, 95(6):L33–L35, sep 2008. ISSN 00063495. doi: 10.1529/biophysj.108.138826. URL https://linkinghub.elsevier.com/retrieve/pii/S0006349508784029.

[226] N.A. N'Dri, W. Shyy, and R. Tran-Son-Tay. Computational Modeling of Cell Adhesion and Movement Using a Continuum-Kinetics Approach. *Biophysical Journal*, 85(4):2273–2286, oct 2003. ISSN 00063495. doi: 10.1016/S0006-3495(03)74652-9. URL https://linkinghub.elsevier.com/retrieve/pii/S0006349503746529.

[227] Ivan Cimrak. *Computational Blood Cell Mechanics*. CRC Press, oct 2018. doi: 10.1201/9781315146775.

[228] M. Dao, C. T. Lim, and S. Suresh. Mechanics of the human red blood cell deformed by optical tweezers. In *Journal of the Mechanics and Physics of Solids*, 2003. doi: 10.1016/j.jmps.2003.09.019.

[229] Thomas M. Fischer, Marianne Stöhr-Liesen, and Holger Schmid-Schönbein. The red cell as a fluid droplet: Tank tread-like motion of the human erythrocyte membrane in shear flow. *Science*, 202(4370):894–896, nov 1978. ISSN 00368075. doi: 10.1126/science.715448.

[230] R. Tran-Son-Tay, S. P. Sutera, and P. R. Rao. Determination of red blood cell membrane viscosity from rheoscopic observations of tank-treading motion. *Biophysical Journal*, 46(1):65–72, jul 1984. ISSN 00063495. doi: 10.1016/S0006-3495(84)83999-5.

[231] Thomas M. Fischer. Shape Memory of Human Red Blood Cells. *Biophysical Journal*, 86(5):3304–3313, may 2004. ISSN 00063495. doi: 10.1016/S0006-3495(04)74378-7.

[232] J. M. Skotheim and T. W. Secomb. Red blood cells and other nonspherical capsules in shear flow: Oscillatory dynamics and the tank-treading-to-tumbling transition. *Physical Review Letters*, 98(7):078301, feb 2007. ISSN 00319007. doi: 10.1103/PhysRevLett.98.078301.

[233] Thomas M. Fischer. Tank-tread frequency of the red cell membrane: Dependence on the viscosity of the suspending medium. *Biophysical Journal*, 2007. ISSN 00063495. doi: 10.1529/biophysj.107.104505.

[234] Ian T. Jolliffe. PRINCIPAL COMPONENT ANALYSIS: A BEGINNER'S GUIDE — I. Introduction and application. *Weather*, 45(10):375–382, oct 1990. ISSN 14778696. doi: 10.1002/j.1477-8696.1990.tb05558.x. URL http://doi.wiley.com/10.1002/j.1477-8696.1990.tb05558.x.

[235] Nvidia. Cuda Toolkit. URL https://docs.nvidia.com/cuda/index.html.

[236] Nvidia. Cuda C Best Practices Guide. *Nvidia Corporation*, 2015.

[237] Philip G. Breen, Christopher N. Foley, Tjarda Boekholt, and Simon Portegies Zwart. Newton vs the machine: solving the chaotic three-body problem using deep neural networks. oct 2019. URL http://arxiv.org/abs/1910.07291.