

Lattice-Boltzmann Simulations of Cerebral Blood Flow

Marco Domenico Mazzeo

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

Department of Chemistry
University College London
May 2009

Published work

*About the most originality that any writer can hope to achieve honestly is to steal
with good judgment*

Josh Billings

I declare that this thesis is the product of my own work, unless otherwise stated, and has not been submitted in any form for another degree or diploma at any university or other institute or tertiary education. It is based in part on work described in the following refereed/to be refereed publications.

1. M. D. Mazzeo, M. Ricci and C. Zannoni, *The Linked Neighbour List (LNL) method for fast off-lattice Monte Carlo simulations of fluids*, Comput. Phys. Commun., **181** (2010) 569.
2. M. D. Mazzeo, S. Manos and P. V. Coveney, *In situ ray tracing and computational steering for interactive blood flow simulation*, Comput. Phys. Commun., **81** (2010) 355.
3. M. Ricci, M. D. Mazzeo, R. Berardi, P. Pasini and C. Zannoni, *A molecular level simulation of a twisted nematic cell*, Faraday Discuss., **144** (2010) 171.
4. R. S. Saksena, B. Boghosian, L. Fajendeiro, O. A. Kenway, S. Manos, M. D. Mazzeo, S. K. Sadiq, J. L. Suter, D. Wright and P. V. Coveney, *Real Science at the Petascale*, Phil. Trans. Royal Soc. A, **367** (2009) 2557.
5. M. D. Mazzeo and P. V. Coveney, *HemeLB: a high performance parallel lattice-Boltzmann code for large scale fluid flow in complex geometries*, Comput. Phys. Commun., **178** (2008) 894.
6. S. K. Sadiq, M. D. Mazzeo, S. J. Zasada, S. Manos, I. Stoica, C. V. Gale, S. J. Watson, P. Kellam, S. Brew, P. V. Coveney, *Patient-specific simulation as a basis for clinical decision-making*, Phil. Trans. R. Soc. A, **366** (2008) 3199.

7. S. Manos, M. D. Mazzeo, O. Kenway, N. T. Karonis, B. Toneen and P. V. Coveney, *Distributed MPI cross-site run performance using MPIg*, Proc. HPDC '08 (2008) 229.
8. S. Manos, S. Zasada, M. D. Mazzeo, R. Haines, G. Doctors, S. Brew, R. Pinning, J. Brooke, Peter V. Coveney, *Patient specific whole cerebral blood flow simulation: A future role in surgical treatment for neurovascular pathologies*, Proc. TeraGrid '08.
9. G. Doctors, M. D. Mazzeo and P. V. Coveney, *A computationally efficient method for simulating fluid flow in elastic pipes in two and three dimensions*, accepted for publication in a journal.

Abstract

Imagination is more important than knowledge...

Albert Einstein

Computational haemodynamics play a central role in the understanding of blood behaviour in the cerebral vasculature, increasing our knowledge in the onset of vascular diseases and their progression, improving diagnosis and ultimately providing better patient prognosis. Computer simulations hold the potential of accurately characterising motion of blood and its interaction with the vessel wall, providing the capability to assess surgical treatments with no danger to the patient. These aspects considerably contribute to better understand of blood circulation processes as well as to augment pre-treatment planning. Existing software environments for treatment planning consist of several stages, each requiring significant user interaction and processing time, significantly limiting their use in clinical scenarios.

The aim of this PhD is to provide clinicians and researchers with a tool to aid in the understanding of human cerebral haemodynamics. This tool employs a high performance fluid solver based on the lattice-Boltzmann method (coined HemeLB), high performance distributed computing and grid computing, and various advanced software applications useful to efficiently set up and run patient-specific simulations. A graphical tool is used to segment the vasculature from patient-specific CT or MR data and configure boundary conditions with ease, creating models of the vasculature in real time. Blood flow visualisation is done in real time using *in situ* rendering techniques implemented within the parallel fluid solver and aided by steering capabilities; these programming strategies allows the clinician to interactively display the simulation results on a local workstation. A separate software application is used to numerically compare simulation results carried out at different spatial resolutions, providing a strategy to approach numerical validation. This developed software and supporting computational infrastructure was used to study various patient-specific intracranial aneurysms with the collaborating interventionalists at the National Hospital for Neurology and Neuroscience (London), using three-dimensional rotational angiography data to define the patient-specific vasculature. Blood flow motion was depicted in detail by the visualisation capabilities, clearly showing vortex fluid flow

features and stress distribution at the inner surface of the aneurysms and their surrounding vasculature. These investigations permitted the clinicians to rapidly assess the risk associated with the growth and rupture of each aneurysm. The ultimate goal of this work is to aid clinical practice with an efficient easy-to-use toolkit for real-time decision support.

To my beloved parents, Giuseppina and Michele, and my closest friends...

Acknowledgments

A wise man's question contains half the answer

Solomon Ibn Gabirol

I begin with the acknowledgments by expressing my gratitude to my first supervisor Prof. Peter Coveney and my second supervisor Prof. Frank Smith of the Mathematical Modeling Group (MMG) who gave me the chance to carry out my PhD, which was funded by an EPSRC studentship allocated by the RealityGrid project. Three months of my PhD course were granted by the EPSRC-funded GENIUS project. I am very lucky to have conducted my scientific activity in such an interesting atmosphere which includes my research group at the Centre of Computational Science (CCS) and London, one of the most fascinating cities in the entire world. I will never forget all the corresponding lessons, stimuli and friendships of the last four years. Thank you very much Peter for supporting me constantly and for assisting my scientific work with precision and rapidity. Thank you Frank for your helpful discussions and for reviewing some of my material.

I am very grateful to my colleagues Steven Manos, Stefan Zasada and Gary Doctors of the Centre of Computational Science (CCS), University College London, for their active and important collaboration within the GENIUS project, which was supported by EPSRC grant EP/F00561X/1, and National Science Foundation MRAC and LRAC awards. Their work was essential and very profitable. I also thank Radhika Saksena, Simon Clifford and Daniel Scott of the CCS for providing me with their technical support, and Luis Fazendeiro for discussions related to the lattice-Boltzmann method.

I thank all CCS, including our project manager Nilufer Betik, for all the nice time spent together and for trying to involve me in extra-curricular activities. I am also grateful to Nilufer for her essential help with bureaucratic issues.

I would like to thank the GENIUS project clinical collaborators for helpful discussions - Dr Stefan Brew, Dr Fergus Robertson, Ann Taylor and Dave Plummer (NHNN, Queens Square, London, UK). I thank very much Shawn Brown at Pittsburgh Supercomputing Center and Kevin Roy, of Cray Research, for conducting benchmarks on Cray XT3 and XT4 platforms respectively. I am very grateful to Brian Toonen

and Prof. Nicholas Karonis of the Mathematics and Computer Science Division at Argonne National Laboratory as well as TeraGrid support for technical assistance and help in setting up cross-site runs. I am grateful to the LONI (Louisiana Optical Network Initiative) staff Jon McLaren, Daniel Katz, Shangli Ou and Michael Lambert, NCSA staff Doru Marcusiu and Tim Cockerill, SDSC staff Kenneth Yoshimoto and Martin Margo, and Jay Boisseau, Lucas Wilson and the support teams of TACC and NGS. I thank Prof. Alfons Hoekstra and Abdel Monim Artoli for their helpful discussions about modelling aspects.

Finally, I am indebted to my family and my dear friends Andrea Protti, Barbara Celli and Simone Rozzi for supporting me constantly and in different ways. In the last four years I have experienced several different issues and their faith, encouragements and support were fundamental. Barbara also reviewed some parts of my thesis. I also thank my other close friends whose names are not listed here but whose importance is not marginal at all, since friendships and social activities play a crucial role in my life, and gave me more or less directly the energy and ambition to complete my PhD.

My family is the best in the world and taught me so many different and fundamental aspects about life. But I have also understood that some friendships are an essential ingredient of my life and will be eternal. Therefore, this thesis is devoted to my family and my closest friends as well.

Contents

Introduction and motivation	21
1 Haemodynamics	24
1.1 Blood circulation	24
1.1.1 Cerebral circulation	24
1.1.2 Blood flow parameters	26
1.2 Cerebrovascular diseases	28
1.2.1 Cerebral atherosclerosis	28
1.2.2 Arteriovenous malformations	28
1.2.3 Aneurysms	29
1.3 Angiography	30
1.3.1 Angiography based on X-rays	32
1.3.2 Magnetic Resonance Imaging	32
1.3.3 Ultrasound	33
1.4 Blood flow modelling	33
1.4.1 Haemodynamic investigations of vascular diseases	34
1.5 Blood flow modelling as a pre-operational treatment planning tool	38
1.6 Summary	39
2 The lattice-Boltzmann method	40
2.1 Computational Fluid Dynamics	40
2.2 The lattice-Boltzmann method	41
2.2.1 Overview	41
2.2.2 The lattice-Boltzmann method and the Lattice Gas Automata technique	43
2.2.3 The lattice-Boltzmann method and the Boltzmann kinetic theory of fluids	43
2.2.4 Details of the lattice-Boltzmann method	44
2.2.5 Derivation of the Navier-Stokes equations	47
2.2.6 Kernel of the lattice-Boltzmann method	49
2.3 Non-standard LB schemes and improvements	51

2.4	Computational aspects	53
2.5	Summary	54
3	The fluid solver, HemeLB	55
3.0.1	Overview	55
3.1	The HemeLB model	58
3.1.1	Boundary condition methods	59
3.2	The HemeLB computational core	63
3.2.1	Preliminaries	63
3.2.2	Handling sparse systems	64
3.3	Parallel implementation	67
3.3.1	Domain decomposition	67
3.3.2	Parallel programming optimisations	70
3.4	Performance scenario	74
3.4.1	Single processor performance	74
3.4.2	Single-machine parallel performance	77
3.4.3	Cross-site parallel performance	79
3.5	Conclusions	80
4	High performance volume rendering and steering of time-varying fluid flow	82
4.1	Introduction	82
4.2	Related work	84
4.2.1	Visualisation of static volumetric datasets	84
4.2.2	Visualisation of time-varying volumetric datasets	86
4.3	The visualisation pipeline	88
4.3.1	Volume rendering core	89
4.3.2	Further optimisations	94
4.3.3	<i>In situ</i> visualisation	96
4.3.4	The compositing scheme	97
4.3.5	Steering approach	98
4.4	Performance results	100
4.4.1	Ray casting performance	100
4.4.2	Parallel ray casting performance and scalability	102
4.4.3	Interactive performance	108
4.5	Conclusions	109
4.6	Appendix: Naïve versus optimised ray traversal engine	110
5	Haemodynamics on computational grids	113
5.1	Overview	114

5.1.1	Previous work	114
5.1.2	Present work	114
5.2	Grid infrastructure	116
5.2.1	Advance reservations	116
5.2.2	Emergency computing	117
5.2.3	The Application Hosting Environment	117
5.2.4	Distributed computing	118
5.2.5	Computational steering	118
5.2.6	Dedicated networking with lightpaths	119
5.3	Medical simulation in the operating theatre	120
5.3.1	Acquiring patient-specific neurovasculatures	121
5.3.2	Data anonymisation	121
5.3.3	The clinical work-flow	121
5.4	Discussion	123
6	Fluid flow simulation in simple and patient-specific systems	124
6.1	Simulations of fluid flow in simple geometries	125
6.1.1	Two-dimensional results	125
6.1.2	Three-dimensional results	130
6.2	Patient-specific cerebral blood flow simulations	137
6.2.1	Patients, images and vascular models	138
6.2.2	Blood flow modelling and visualisation	140
6.3	Conclusion	146
	Conclusion	151
	Bibliography	155

List of Figures

1.1	Schematic illustration of the Circle of Willis.	25
1.2	A typical pressure waveform at the internal carotid artery. Image taken from [1].	26
1.3	Blood pressure as a function of arterial vessel calibre. This illustration provides an indicative representation of the pulsatile pressure waveform as a function of artery type. Image taken from [2].	27
1.4	Volumetric flow rate (in ml/s) waveform for three pulsatile cycles within the right internal carotid artery measured by using phase contrast magnetic resonance angiography, discussed in Sec. 1.3.2. Image taken from [3].	27
1.5	Normal vascular configuration (left) and AVM (right). Note that the present AVM has a large (abnormal) vascular link ("fistula") in the centre while normal capillaries are around it. Figure taken from a presentation by Dr. Stefan Brew, National Hospital for Neurology and Neuroscience, Queen Square, London.	28
1.6	A saccular (or bubble) aneurysm and a fusiform aneurysm on the left and right hand side respectively.	30
1.7	Aneurysm clipping. Image taken from [4].	30
1.8	Aneurysm coiling. Image taken from [4].	31
1.9	A stent redirects the flow through the parent artery. Image taken from [4].	31
2.1	Schematic illustration of the advection step of the lattice-Boltzmann method (see text for details on the symbols).	44
2.2	Schematic illustration of the computational work-flow of a LBGK method (see text for details on the symbols).	50
3.1	Schematic diagram of a smooth boundary for which both horizontal inward-pointing directions indicated by the arrows originate from solid sites. Lattice sites within the computational fluid domain are denoted by empty circles. Many boundary condition methods are unable to determine the unknown distribution functions at the central fluid lattice site.	59

3.2	Two-dimensional representation of the two-level hierarchical data organisation of an idealised bifurcation (thick dashed edges). The bounding box comprises 4×4 macro-cells. Its fluid and solid lattice sites of the Cartesian computational grid are depicted by means of voxels with solid and dashed edges respectively. Only the data associated with each non-empty macro-cell are allocated in memory.	64
3.3	Two-dimensional illustration of the two-level data layout (left image) set up and used during the pre-processing stage only, and reordered data format of f_{source} or f_{dest} (right image) employed by HemeLB during the simulation. It is worth noting that data related to different voxel types are kept separate in f_{source} and f_{dest} . See text for further details.	66
3.4	Two-dimensional illustration of the domain partitioning algorithm taking place within a voxelised bifurcation. The first processor sub-domain (voxels covered by black arrows) is created by expanding a volume region of fluid voxels through the computational lattice from the left- and bottom-most fluid voxel. When N/P (N and P are the number of fluid voxels and processors respectively) fluid voxels are assigned to the first processor, a new propagation through the lattice, which will generate the sub-domain to be assigned to the second processor, starts from another fluid voxel (red arrows), and so on. See text for further details.	67
3.5	Two-dimensional partitions of a square and a bifurcation with 128^2 and 61451 fluid lattice sites respectively, as obtained with the domain decomposition method of Wang <i>et al.</i> [5] (left hand side images) and that presented in Sec. 3.3.1 (right hand side). In each image the number of partitions is 16 and their ranks are coloured according to a grey-scale map. The identifiers of the interface-dependent distribution functions between every pair of adjacent partitions are represented by tiny grey segments. The grey intensity is proportional to the identifier. Adjacent distribution functions have the same identifier and cannot be communicated during each time step.	70
3.6	External pressure profile of the stationary flow field pertaining to a bifurcation of 6253409 fluid lattice sites. The wall and pressure boundary conditions applied to the no-slip walls and inlet/outlet lattice sites respectively are discussed in Sec. 3.1. Red and blue colours represent maximum and minimum pressure respectively. The colour of any fluid lattice site is linearly interpolated between those values according to its pressure.	75

3.7	Performance measured in millions of fluid site updates per second (MSUPS) as a function of the cube root of the fluid lattice sites using HemeLB and a HemeLB's version ('direct two-level', see Sec. 3.2 for details) which explicitly adopts the two-level data representation of HemeLB and therefore is cache-aware but does not exploit the other optimisations presented in Sec. 3.2. The timing results were conducted on a single core of a Intel Core 2 Quad 2.5 GHz with 3 MB cache/core running the operating system Linux Fedora 8 64 bits; the compiler used was Intel 9.1 with flags <code>-O3, -ipo, -xT</code>	76
3.8	External pressure profile of the stationary flow field pertaining to the benchmarked patient-specific system which comprises 7.7M fluid lattice sites. Red and blue colours represent maximum and minimum pressure respectively. The colour of any fluid lattice site is linearly interpolated between those values according to its pressure.	78
3.9	HemeLB's single-site parallel performance measured in millions of lattice site updates per second (MSUPS) as a function of the number of processor cores used for the patient-specific system of 7.7M fluid lattice sites on platforms HPCx and Abe (see Sec. 3.4.2 for further details). Ideal performance is depicted with dashed lines.	79
3.10	HemeLB cross-site performance in millions of fluid lattice site updates per second (MSUPS) as a function of the number of LONI clusters, each contributing 32 processor cores (see Sec. 3.4.3 for further details). Ideal performance is depicted with a dashed line. The fluid flow comprises 4.69 fluid lattice sites. MPIg was employed to handle inter-machine communications. The executable was obtained compiling with flags <code>-O2, -qstrict -qstrict_induction</code>	80

- 4.1 Two-dimensional representation of a ray casting approach to handle volume rendering of a voxelised birfurcating 3D dataset; (a) standard technique (left image) and (b) our method (three rightmost images) which decomposes this example system into three volume clusters (see Sec. 4.3.1 for details). The transparent and opaque voxels of the two-level grid used to accelerate ray traversal are represented with dashed borderlines and black squares respectively. Non-empty macrocells comprise 4×4 voxels. The computational cost of a ray is proportional to its thickness: a ray that is only generated and is tested against a bounding box for intersection purposes has the minimum thickness, while a ray which also traverses empty and/or full macrocells is thicker. The total number of ray generations and ray-box intersections is $4 + 3 + 6 = 13$ with our method (instead of 16); here, ray traversal within some macrocells and voxels is avoided because the bounding boxes of the subdomains are shrunk with respect to the original system (thus empty macrocells are not taken into account). Conversely, our method considers post-first-hit ray traversal steps which can be avoided in the conventional technique: our algorithm needs to assemble sub-images pertaining to different subdomains in a single image. 92
- 4.2 Illustration of our binary tree communication pattern over time to handle image compositing using 9 processors numbered 0 to 8. Each processor sends its sub-image to a neighbouring one with lower rank, specifically depending on the tree level. The thickness of each arrow denotes the message size, while the one of each line represents the time to merge the received sub-image. At the end of the process the 0-rank processor has the final image which can be written to file or communicated to a remote computer. Load imbalance can arise, which is a result of the binary tree communication pattern used as well as different sub-image sizes. 98

4.3	Temporal sketch of our interactive simulation environment. Depending on the pipeline stage from top to bottom, the number in each box refers to the simulation time step currently processed, rendered or displayed, and finally explored. The simulation and volume rendering take place on the same platform, while the image of the flow field is transferred to a remote computer client and displayed. The clinician can explore and interact with the simulation output by interactively using the steering capabilities. One advantage of the scheme used is that the fluid solver is not substantially affected by the speed of the other environment components (see Sec. 4.3.5 and 4.4 for further details and performance results respectively).	99
4.4	From left to right, we show the first, second and third models considered in this work (referred to as Model 1, Model 2 and Model 3). They correspond to three highly-resolved digitally reconstructed patient-specific intracranial vascular pathologies and comprise 1661545, 183228 and 249094 sparsely distributed fluid voxels respectively. In the three images, we depict 64 colour-coded subdomains. Each subdomain is subdivided into one or more volume clusters which are rendered independently by tracing rays through their minimal viewport rectangles (see Sec 4.3.1 for details). The boxes are outlined in black.	100
4.5	The top-left image shows the pressure distribution at the wall of Model 1, a large aneurysm comprised of 1661545 fluid voxels, while the bottom-left image depicts the part of the vasculature characterised by a velocity greater than 20% of the maximum value. The colour scale depicts low (violet) values to high (red) values. The plots on the right hand side report the performance achieved on a single Intel quad-core 2.5 GHz with a viewport of 800^2 pixels using various volume rendering approaches as a function of the number of partitions in which every vasculature is subdivided. Specifically, the horizontal dashed line is the single-core performance achieved by the explicit kd-tree. We applied the object-based technique presented in Sec. 4.3.1 to the ray caster based both on our bi-level grid and the explicit kd-tree; their performances are also shown. See Sec. 4.4.1 for further details.	103

4.6	The top-left image shows the pressure distribution at the wall of Model 2, a small aneurysm comprised of 183228 fluid voxels, while the bottom-left image depicts the part of the vasculature characterised by a velocity greater than 20% of the maximum value. The colour scale depicts low (violet) values to high (red) values. The plots on the right hand side report the performance achieved on a single Intel quad-core 2.5 GHz with a viewport of 800 ² pixels using various volume rendering approaches as a function of the number of partitions in which every vasculature is subdivided. See Sec. 4.4.1 for further details.	104
4.7	The top-left image shows the pressure distribution at the wall of Model 3, a small aneurysm comprised of 249094 fluid voxels, while the bottom-left image depicts the part of the vasculature characterised by a velocity greater than 20% of the maximum value. The colour scale depicts low (violet) values to high (red) values. The plots on the right hand side report the performance achieved on a single Intel quad-core 2.5 GHz with a viewport of 800 ² pixels using various volume rendering approaches as a function of the number of partitions in which every vasculature is subdivided. See Sec. 4.4.1 for further details.	105
4.8	Snapshots of the simulation results obtained with our <i>in situ</i> visualisation system. The three different sets of four images correspond to three different types of vascular pathologies which are Models 1 (top-left set), 2 (top-right set) and 3 (bottom set). For each model, the volume rendering of the velocity and stress flow fields of the flow half way through a pulsatile cycle are depicted on the top-left and top-right images respectively. The wall pressure and wall stresses are shown in the bottom-left and bottom-right images respectively.	106
4.9	Parallel scalability and performance of the <i>in situ</i> rendering system for the vascular pathologies shown in Fig. 4.8, namely Model 1, 2 and 3. The flow field is simulated using the parallel fluid solver HemeLB and the volume rendering is performed by the <i>in situ</i> object-based ray caster built into HemeLB (Sec. 4.3.1). Performance of the fluid solver alone (FS) and the fluid solver with volume rendering (FS+VR) are shown as time steps per second.	107
5.1	GENIUS GUI screenshots. The GENIUS GUI co-reserves a certain number of hours with HARC (top left), generates the computational model (top right), launches the corresponding simulation (bottom left) and interactively monitors and steers HemeLB (bottom right).	119

5.2	The GENIUS lightpath network. Dedicated gigabit links are used to connect computational infrastructure such as the TeraGrid and LONI directly to the NHNN angiography suite. In the case of the NGS2 Manchester and Oxford nodes, a dedicated link is used to facilitate cross-site MPIg runs, with the NGS2 Manchester → UCL link used for steering, visualisation and medical data transfer.	120
5.3	Our clinical workflow. The patient-specific dataset is acquired through a medical scanner. Then, the dataset is segmented and manipulated to impose blood flow conditions at the boundaries of the selected region of interest. The corresponding data are migrated to a supercomputer where the blood behaviour is simulated and rendered during a HARC pre-reserved slot. The simulation is interactively monitored and controlled by means of the steering capabilities implemented in HemeLB and our graphical user interface (see 5.3.3 for further details).	122
6.1	Two-dimensional illustration of a lattice within a channel of size (L_x, L_y) (L_y is the height) whose fluid flow is confined by the horizontal no-slip walls and is driven by the pressure-controlled vertical boundaries. The boundary sites are denoted by empty circles. Note that the total fluid lattice sites are $(L_x + 1) \times (L_y + 1)$	126
6.2	Two-dimensional illustration of a lattice within a channel of size (L_x, L_y) (L_y is the height) whose fluid flow is confined by the horizontal no-slip walls and is driven by the pressure-controlled vertical boundaries. The lattice sites close to the boundaries are denoted by empty circles. Note that the total fluid lattice sites are $(L_x - 1) \times (L_y - 1)$ if $q_u = q_p = 1$. At the bottom, we show the approximations employed by Guo <i>et al.</i> [6] to approach pressure-controlled boundaries (see text for further details).	132
6.3	Snapshot of the graphical editing tool used to segment vasculatures and configure inflow/outflow boundary conditions. Several functions, as listed on the menu on the left hand side, are employed to configure boundary geometry and parameters. Here, we can see two boundaries (grey and green triangles) and the cropped vasculature therein.	139

6.4	Eight snapshots subdivided in two panels with four quadrants each obtained at the diastolic (uppermost panel) and systolic (lowermost panel) pressure peaks of the aneurysm model M_1 obtained at the highest spatio-temporal resolution (see Sec. 6.2.2 for details). For each panel, the volume rendering of the velocity and von Mises stress flow fields are displayed within the top left and top right quadrants respectively; the bottom right and bottom left ones show the pressure and von Mises stress wall distributions, and particle traces (streaklines).	147
6.5	Eight snapshots subdivided in two panels with four quadrants each obtained at the diastolic (uppermost panel) and systolic (lowermost panel) pressure peaks of the aneurysm model M_2 obtained at the highest spatio-temporal resolution (see Sec. 6.2.2 for details). For each panel, the volume rendering of the velocity and von Mises stress flow fields are displayed at the top left and top right quadrants respectively; the bottom right and bottom left ones show the pressure and von Mises stress wall distributions, and particle traces (streaklines).	148
6.6	Eight snapshots subdivided in two panels with four quadrants each obtained at the diastolic (uppermost panel) and systolic (lowermost panel) pressure peaks of the aneurysm model M_3 obtained at the highest spatio-temporal resolution (see Sec. 6.2.2 for details). For each panel, the volume rendering of the velocity and von Mises stress flow fields are displayed at the top left and top right quadrants respectively; the bottom right and bottom left ones show the pressure and von Mises stress wall distributions, and particle traces (streaklines).	149

List of Tables

4.1	Performance results for interactive steering with the display outputs for the three datasets used; their snapshots are shown in Fig. 4.8 when the image size is large.	108
6.1	Errors, orders of accuracy and number of time steps (ts) for the two-dimensional Poiseuille flow simulated with the LBGK models coined D2Q9 and D2Q9i at $Re = 1.28$ with $u_0 = 0.064\Delta x$ and $\nu = 0.05$. “MA” means that accuracy similar to the machine precision is achieved. See text for further details.	128
6.2	Errors, orders of accuracy and number of time steps (ts) for the two-dimensional Poiseuille flow simulated with the LBGK models coined D2Q9 and D2Q9i at $Re = 12.8$ with $u_0 = 0.64\Delta x$ and $\nu = 0.05$. “MA” means that accuracy similar to the machine precision is achieved. See text for further details.	128
6.3	Errors and number of time steps (ts) for the two-dimensional Poiseuille flow simulated with the LBGK model coined D2Q9 with $\nu = 0.2\Delta x$. See text for further details.	128
6.4	Errors and number of time cycles for the two-dimensional time-varying fluid flowing through a rectilinear channel simulated with the LBGK model coined D2Q9. $\nu = 400\Delta t$ while $u_{max} = 8.3\Delta t$, $u_{max} = 83\Delta t$ and $u_{max} = 830\Delta t$ for Re equal to 0.664, 6.64 and 66.4 respectively. See text for further details.	130
6.5	Errors, orders of accuracy and number of time steps (ts) for the fluid flow within a square duct and simulated with the LBGK model coined D3Q15, $Re = 0.754$, maximal velocity $u_{max} = 0.03772\Delta x$ and $\nu = 0.05$. See text for further details.	135
6.6	Errors, orders of accuracy and number of time steps (ts) for the fluid flow within a cylinder inclined by polar and vertical angles $\theta = \pi/3$ and $\phi = 2\pi/9$ respectively and simulated with the LBGK model coined D3Q15, $Re = 0.64$, maximal velocity $u_{max} = 0.032\Delta x$ and $\nu = 0.05$. See text for further details.	136

6.7	Geometrical information of the vascular patient-specific models investigated in this work; “small” and “large” mean about 5 mm and slightly more than 10 mm respectively. MCA and ICA stands for middle cerebral artery and internal carotid artery respectively.	140
6.8	Blood flow results pertaining to the vascular patient-specific aneurysms M_1 , M_2 and M_3 investigated in this work. Velocity and stress values are expressed in m/s and Pa respectively. s_{max} and v_{max} are the maximum values of the von Mises stress (see Chapter 2, Eqn. 2.20) and of the velocity magnitude respectively, $\langle v_i \rangle$ and v_i^{max} are the average and peak inflow velocity magnitudes respectively. “voxels” and “cycles” are the number of fluid sites and cardiac cycles to achieve numerical convergence respectively. See text for further details.	143
6.9	Average L_1 -errors in percentage calculated by considering the difference in the pressure, velocity and von Mises stress flow fields of the three aneurysm patient-specific models M_1 , M_2 and M_3 of Table 6.7 carried out with the two lowest spatio-temporal resolutions compared with the results of the highest one. See text for further details.	144

Summary

The true teacher defends his pupils against his own personal influence

Amos Bronson Alcott

Cardiovascular disease is the cause of a large number of deaths in the developed world [7]. The understanding of the blood behaviour in the vascular system, *i.e.* haemodynamics, and its interaction with the tissues of the vasculature plays a crucial role in the knowledge of the genesis, progression, diagnosis and treatment of vascular diseases, such as aneurysms, arteriovenous malformations and atherosclerosis.

Patient-specific medicine is the tailoring of medical treatments based on the characteristics of an individual patient. Since patient-specific data can be used as the basis of simulation, treatments can be assessed for effectiveness with respect to the patient in question before being administered, giving clinicians more information to make their decisions. For example, planned surgery of an aneurysm could benefit from a better understanding of the processes related to its formation, progression and rupture, which are not well understood yet.

In the last fifteen years, computational haemodynamics has played a key role in the understanding of haemodynamic phenomena and the improvement of medical treatment.

This work is intended to aid the understanding of human intra-cranial patient-specific haemodynamics through the use of efficient software consisting of a parallel fluid solver, high performance computing and grid computing, and a number of advanced software applications which support effective patient-specific setups and investigations. The ultimate goal of this work is to aid clinical practice with an efficient and easy-to-use toolkit for real-time decision support.

This thesis begins with an overview of blood circulation, cerebrovascular diseases and imaging techniques used nowadays for diagnostic purposes, and computational approaches which accurately characterise blood flow. In particular, some of the works discussed at the end of the first chapter provide evidence of the correlation between specific haemodynamic factors and vascular pathology progression, such as the effect of a very low wall shear stress on the wall remodelling and aneurysm growth. The employment of the simulation tool has played a crucial role in this context. In Chap-

ter 6, key points related to those studies and discoveries are taken into account to qualitatively assess rupture risk of malformed patient-specific cerebrovasculatures.

The simulation technique employed in this work is the lattice-Boltzmann method which is the subject of the second chapter. Here, theoretical and practical aspects of the lattice-Boltzmann method are described. Furthermore, the main advantages and limitations of the lattice-Boltzmann method with respect to other techniques adopted to describe fluid flow dynamics are discussed.

In this work, a high performance lattice-Boltzmann blood flow simulator, called HemeLB, has been developed. Its description is the topic of Chapter 3; specifically, the technical and performance aspects of HemeLB are outlined in detail.

The fourth chapter begins with a literature review concerning efficient visualisation of static and time-varying fluid flow simulations. Then, the visualisation technique developed and exploited in this work, and a discussion about its advantages and limitations are presented. Specifically, our visualisation method adopts an efficient *in situ* fluid flow rendering approach, incorporated in HemeLB, to effectively provide visual feedback of the simulation results. The corresponding software application has been enhanced with steering capabilities, built into HemeLB, to permit interactive exploration, steering and analysis of the simulation.

An EPSRC-funded project called GENIUS (Grid Enabled Neurosurgical Imaging Using Simulation) started during my PhD research with the aim of creating a grid-based interactive simulation environment and integrate it into the clinical practice to aid the knowledge of the genesis, progression, diagnosis and treatment of vasculature diseases. The fifth chapter focusses on the grid infrastructure developed within the GENIUS project. The simulation, visualisation and steering capabilities of HemeLB played a fundamental role in this, but the fifth chapter discusses other aspects important for the success of the GENIUS project, such as advance reservations, emergency computing, the anonymisation of patient data and the seamless integration and usage of all the components of the GENIUS middleware.

Chapter six firstly presents various fluid flow results of rectilinear geometries to validate different boundary condition methods and the model specifically used in HemeLB. Then, patient-specific blood flow models using different simulation setups are presented. Here, various haemodynamic aspects related to three aneurysms are discussed in detail to draw conclusions about their rupture risk. Furthermore, the efficacy of our interactive simulation environment is demonstrated.

The thesis ends with a brief summary, concluding remarks and a discussion about future work.

Chapter 1

Haemodynamics

Teachers open the door. You enter by yourself.

Chinese Proverb

The interplay between haemodynamics and the mechanical and biological phenomena associated with the wall of the vessels plays a crucial role in the diagnosis, progression and treatment of vascular diseases.

This chapter provides a general overview of haemodynamics with some emphasis on cerebral circulation and malformations, and outlines several aspects and studies conducted in the last fifteen years to enhance the understanding of the pivotal points concerning vascular pathologies, such as the interplay between their formation or progression and blood flow behaviour.

1.1 Blood circulation

Blood is transported through the complex network of vessels, which form the cardiovascular system. In general, blood behaves like a Newtonian fluid in large vessels [8], *i.e.* with diameter greater than 1 mm. The relationship between the shear rate and shear stress of the fluid is linear and is proportional to the viscosity. In smaller vessels, such as the capillaries, the blood behaviour is non-Newtonian [8].

The blood flow through the cardiovascular system is driven by the action of the heart, which yields a pressure difference between the aorta and the veins. The blood passes from the capillaries where the dissolved gasses and metabolites are exchanged by diffusion; the resulting deoxygenated blood then returns to the heart via the veins.

1.1.1 Cerebral circulation

The brain is supplied with blood through the two vertebral arteries and the two internal carotid arteries (ICAs), accounts for 15 % of total cardiac output and consumes 25 % of the oxygen, which is substantial. The vertebral arteries join to form the

basilar artery. The basilar artery and the ICAs fuse together to form the Circle of Willis (CoW). The basilar artery has two branches, the left and right superior cerebral arteries (SCAs). The basilar artery branches into the left and right Posterior Cerebral Arteries (PCAs) which are linked to the ICAs through the Posterior Communicating arteries (PcomAs). The PcomAs divide the PCAs into the P1 and P2 segments. The ICAs become the Middle Cerebral arteries (MCAs) and the Anterior Cerebral Arteries (ACAs) branch from them. The anterior communicating artery (AcomA) connects the ACAs and divides them into the A1 and A2 segments. The anatomy of this structure is illustrated in Fig. 1.1.

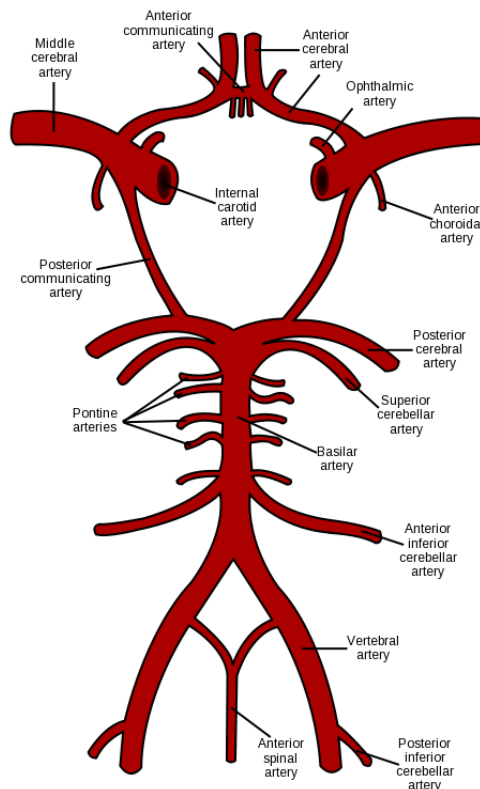


Figure 1.1: Schematic illustration of the Circle of Willis.

The CoW usually allows the blood flow to be maintained even if one of the arteries becomes blocked. Many people, in fact, do not experience serious cerebral problems even if one of the important arterial links does not exist. Variations in the CoW include one or more absent vessels, abnormally thin vessels and duplication or triplication of vessels.

Blood passes from the veins called dural venous sinuses which drain into the internal jugular vein. One of the largest veins is the Great Cerebral Vein, or vein of Galen.

Mean arterial pressure is typically between 60 and 140 mm·Hg. Cerebral vessels have the ability to locally maintain a constant blood flow despite drastic pressure alterations such as those following a vessel rupture or a vessel obstruction. For example, if pressure is decreased by partially occluding an artery, blood flow initially falls,

then returns towards normal levels over the next few minutes. Conversely, increases in the mean arterial blood pressure lead to constriction of the vessels which in turn reduces blood flow. The autoregulation mechanism is intrinsic to the vessels because it is not subjected to neural and hormonal influences.

1.1.2 Blood flow parameters

The speed of sound within blood depends on Haematocrit¹, temperature and frequency and is between 1500 m/s and 1600 m/s while its viscosity substantially depends on Haematocrit and shear rate [9]. Any part of the blood that is not a red blood cell is occupied by plasma, whose density depends only on temperature and is between 1000 kgm^{-3} and 1100 kgm^{-3} . The viscosity depends on many factors, but is fairly constant at high shear rates. A pulsatile period under normal physical conditions is close to 1 s. The pressure at the ICA or basilar artery is often taken to be periodic, with a systolic pressure of 120 mm·Hg and a diastolic pressure of 80 mm·Hg [1,10,11]. The waveform is shown in Fig. 1.2 [1]. The blood pressure gradually decreases and becomes less pulsatile further down the arterial tree, as shown in Fig. 1.3. The pressure waveforms of the ICAs are typically in phase, while they may lag the flow at the vertebral arteries by 0.025 s or less [12]. A typical flow rate waveform of the right ICA is shown in Fig. 1.4. The flow rates in each ICA and vertebral artery are approximately 4.8 ml/s and 0.7 ml/s respectively; the venous pressure is between 5 mm·Hg and 20 mm·Hg [13].

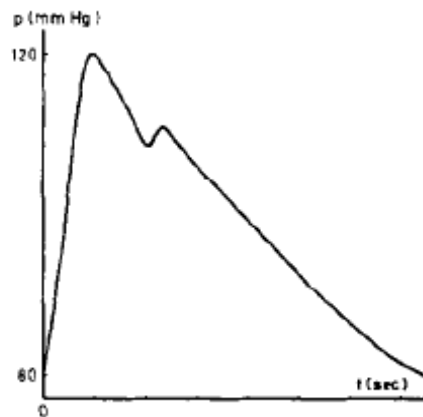


Figure 1.2: A typical pressure waveform at the internal carotid artery. Image taken from [1].

¹Proportion of blood volume that is occupied by red blood cells.

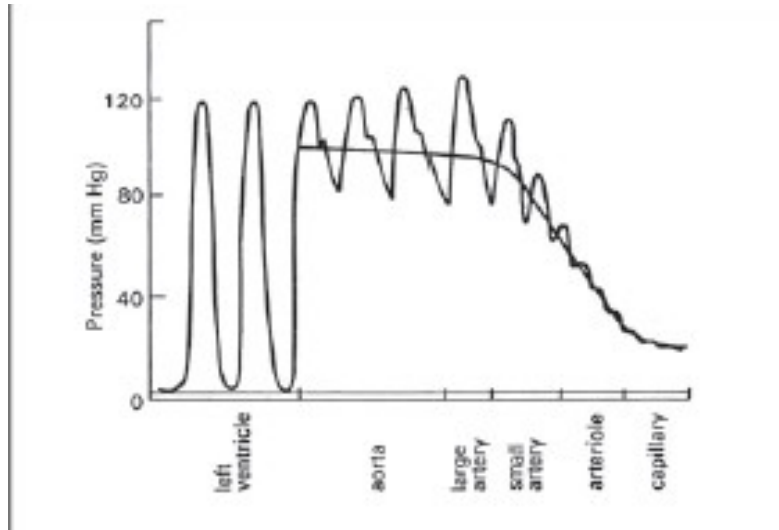


Figure 1.3: Blood pressure as a function of arterial vessel calibre. This illustration provides an indicative representation of the pulsatile pressure waveform as a function of artery type. Image taken from [2].

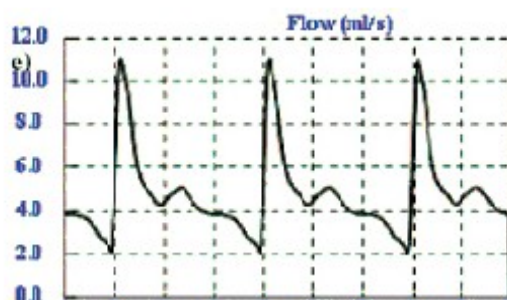


Figure 1.4: Volumetric flow rate (in ml/s) waveform for three pulsatile cycles within the right internal carotid artery measured by using phase contrast magnetic resonance angiography, discussed in Sec. 1.3.2. Image taken from [3].

1.2 Cerebrovascular diseases

Within this subsection, cerebrovascular diseases, such as cerebral atherosclerosis, cerebral aneurysm and arteriovenous malformation, are discussed in detail.

1.2.1 Cerebral atherosclerosis

Atherosclerosis is characterised by the presence of a plaque on the internal layer of an artery. This plaque has the potential to rupture, or to occlude the affected artery through the formation of a blood clot (thrombus). As a consequence, atherosclerosis may have a great influence on the blood flow behaviour and the blood supply to vital organs. For instance, atherosclerosis may provoke ischaemia, a reduction in blood supply which destroys the brain tissue, or a stroke which in turn can lead to haemorrhaging (bleeding).

In the case of thrombus, thrombolytic agents or anticoagulants may help to remove the clot. Other therapies include the reduction of factors which have high risk associated with atherosclerosis, such as hypertension, diabetes and smoking. Alternatively, an arterial deposit can be surgically removed or treated by angioplasty, which is the widening of a vessel by using folded balloons followed by a stent placement. Finally, a bypass can supply enough blood between the sections of the artery surrounding the atherosclerosis.

1.2.2 Arteriovenous malformations

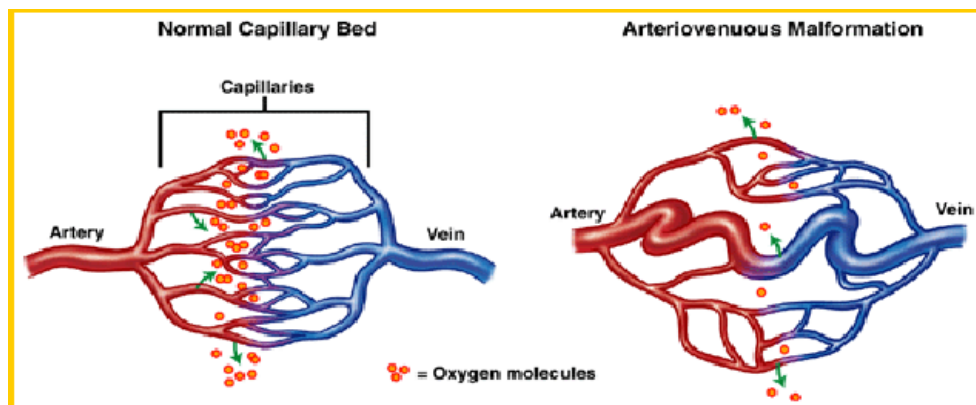


Figure 1.5: Normal vascular configuration (left) and AVM (right). Note that the present AVM has a large (abnormal) vascular link ("fistula") in the centre while normal capillaries are around it. Figure taken from a presentation by Dr. Stefan Brew, National Hospital for Neurology and Neuroscience, Queen Square, London.

An arterio-venous malformation (AVM) is an abnormal connection between veins and arteries, as shown in Fig. 1.5. Structurally, arteries divide and sub-divide repeatedly, eventually forming a sponge-like capillary bed. An AVM lacks the high resistance

of capillaries on the blood flow; it also causes the surrounding area to be deprived of the functions of the capillaries *e.g.* removal of CO₂ and delivery of nutrients to the cells. The resulting tangle of blood vessels, often called a *nidus* has no capillaries and abnormally direct connections between high-pressure arteries and low-pressure veins. The small vessels of the *nidus* are prone to bleeding because their muscle layer is deficient. One type of AVM is the vein of Galen aneurysm malformation (VGAM). Treatment ranges from surgical resectioning, radiosurgery, embolisation to a combination of these; embolisation is the occlusion of blood flow with a glue deposited upstream of the nidus by the use of endovascular catheters. The aim of treatment is the complete removal of the AVM, since partial removal does not mitigate the risk of haemorrhage. Surgical resectioning offers an immediate solution but is risky.

1.2.3 Aneurysms

An aneurysm is a localized dilation of a vessel caused by disease or weakening of the vessel wall. Aneurysms most commonly occur in arteries at the CoW and in the aorta. The bulge in a blood vessel can burst and lead to death at any time. The larger an aneurysm becomes, the more likely it is to burst. Given enough time the aneurysms which grow will inevitably experience rupture if not treated and if the aneurysm body does not experience blood clotting so as to regulate the local circulation to a normal behaviour. However, most aneurysms are small and around 50–80 % of all cerebral aneurysms do not rupture during the course of the patient's lifetime.

Aneurysm formation is due to a reduction in the thickness of the vessel wall whose structural defects combined with haemodynamic forces cause that wall to swell.

The danger associated with aneurysm rupture is proportional to its size and aspect ratio, and to some haemodynamic patterns (see Sec 1.4.1). An aneurysm is classified as being large or giant if its size is between 1 cm and 2.5 cm or greater than 2.5 cm respectively. An aneurysm may be fusiform or saccular, in which case it resembles a bubble (Fig. 1.6) [14].

Intracranial aneurysms are treated through clipping, coiling, embolisation or the application of a stent. During clipping a permanent metal clip is located at the neck of the aneurysm as in Fig. 1.7. Clipping can be incomplete or yield recurrence and haemorrhage.

Aneurysm coiling is carried out by using a catheter to position a coil within the aneurysm. The coil should fill the volume of the aneurysm; as a consequence, blood clots within the aneurysm and flows outside. Coiling is illustrated in Fig. 1.8. Risks to the patient associated with coiling include infections, haematomas, occlusion of the parent artery, thromboembolic phenomena and rupture of the aneurysm.

A stent could be used to redirect the blood flow through the parent artery and not the aneurysm (Fig. 1.9); the corresponding alteration in blood flow is facilitated

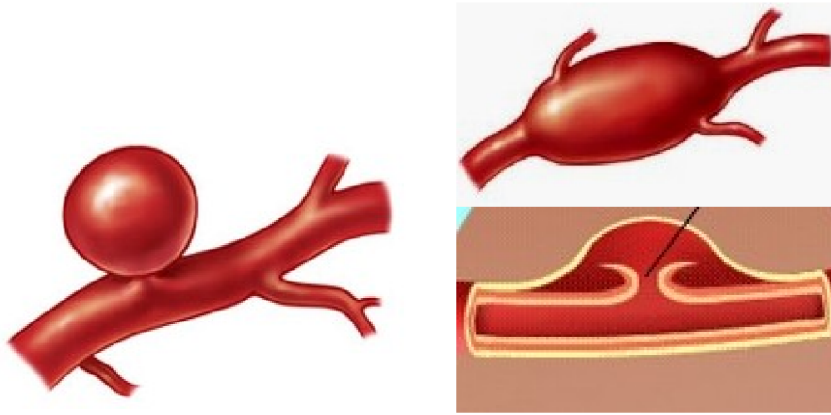


Figure 1.6: A saccular (or bubble) aneurysm and a fusiform aneurysm on the left and right hand side respectively.

if a blood clot forms on the spaces between the struts of the stent, which may occur. Alternatively, the parent artery may be occluded by injecting glue.

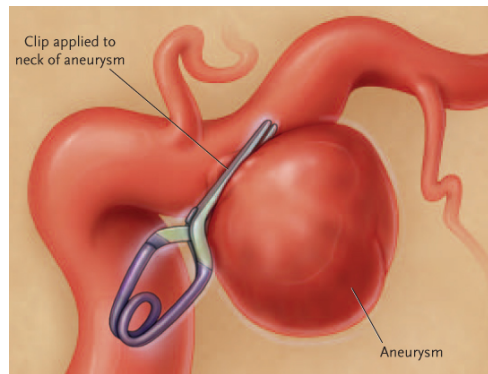


Figure 1.7: Aneurysm clipping. Image taken from [4].

1.3 Angiography

Several imaging techniques are used nowadays to visualise the inside, or lumen, of blood vessels. A medical imaging technique of this type is called "angiography". Some of them are also employed to extract haemodynamic information, such as fluid velocity and flow rate at some locations within the vessels. A non-invasive imaging method is not risky and thereby more feasible and attractive than an invasive approach. This argument is emphasised by the prospect to integrate blood flow simulation into the clinician workflow and to assist doctors in the treatment of vascular pathologies (see Sec. 1.5). The more important aspects of imaging techniques discussed here concern the resolution and contrast which are crucial to perform reliable blood flow modelling. Specifically, the employment of an accurate blood flow solver is important

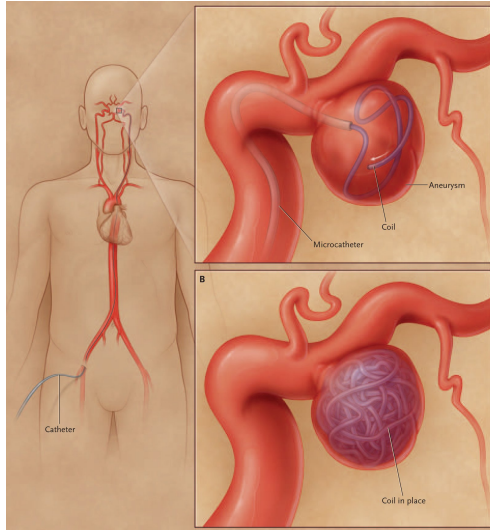


Figure 1.8: Aneurysm coiling. Image taken from [4].

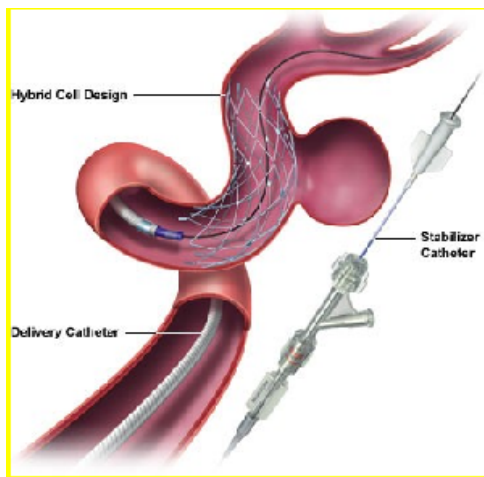


Figure 1.9: A stent redirects the flow through the parent artery. Image taken from [4].

for numerical investigation purposes but a precise geometrical representation of the vessel walls is the first step towards a fully faithful simulation of the blood flow therein (see Sec 1.4.1). Unfortunately, the spatial resolution of the most accurate imaging techniques (0.5 mm) is not much higher than the typical diameter of the main cerebral arteries (1-5 mm). However, it seems that blood flow simulation results pertaining to poor geometrical models do not qualitatively differ from those achieved by accurately defining the vasculature wall [15].

1.3.1 Angiography based on X-rays

During data acquisition through X-ray angiography, two scanners emit X-rays towards the system under investigation along two (ideally orthogonal) projections. A contrast agent is injected into the vascular tree to enhance the interaction between X-rays and blood. Then, the resulting image slices are digitally subtracted from a pre-injection scan to remove the presence of static material. This requires a double dose of radiation and is called Digital Subtraction angiography (DSA). The contrast and spatio-temporal resolutions are better than those obtained via Magnetic Resonance angiography (MRA) and Computer Tomography (CT) scans [16] (see below); unfortunately, catheterisation and adverse reaction to the contrast agent make X-ray angiography a significantly invasive technique. Modern equipment usually provides an isotropic resolution of 0.2–0.3 mm.

A recent development is three-dimensional rotational angiography (3DRA) [14,17], for which the volumetric information is obtained through the acquisition of a series of two-dimensional rotational projections at different angles. Unfortunately, 3DRA and DSA are unable to visualize an entire lesion fed by more than one artery if a contrast agent does not simultaneously pass through both ICAs (which entails performing two simultaneous injections). Alternatively, the contrast agent may be injected into the aortic arch, but this causes a serious patient risk. Recently, Castro *et al.* [18] have reported a computational approach which merges the two sides of the vascular tree.

Finally, computer tomography angiography (CTA) uses X-rays to produce the images but, unlike X-ray angiography, the brain is scanned in slices.

1.3.2 Magnetic Resonance Imaging

During magnetic resonance imaging (MRI), contrast is obtained by measuring, under a static magnetic field, the difference in the magnetic spin relaxation properties of the scanned materials. MRI is usually characterised by an inferior spatial resolution with respect to techniques based on X-rays. However, MR angiograms can be enhanced with paramagnetic contrast agents. Usually, the volumetric resolution is 0.4–1 mm.

A technique known as phase contrast MRI (PC-MRI) relies on appropriately vary-

ing the magnetic field and can provide two and occasionally three-dimensional images of one or more velocity components in a time-resolved manner [16]. For instance, Wetzel *et al.* [19] evaluated this technique and quoted that the resolution was 0.05 s in time and 1.5 mm in space; the scan took 15–20 minutes and the entire processing time was about one hour. This data acquisition offers the prospect to probe the time-varying velocity flow field of a real vasculature by avoiding to introduce error sources connected to the assumptions which are typically made by simulation approaches; for example, numerical blood flow predictions are usually based on assumptions like rigid walls and Newtonian flow and are influenced by the approximate computerised representation of the vasculature. However, the resolution of the blood flow obtained with MRI is not sufficient to calculate and visualise secondary flow parameters, such as the wall shear stress (WSS).

PC-MRA is usually used in image-based computational fluid dynamics (CFD) applications for providing flow-rate waveforms at some planar locations [3] like boundary conditions, which allows cerebral circulation simulations to be more subject-specific. For instance, with four optimally placed imaging planes, the boundary conditions at all afferent and efferent arteries to the CoW can be specified [3]. The accurate measurement of pressure, instead, involves the placement of a transducer inserted via a catheter, which is invasive and dangerous for the patient.

1.3.3 Ultrasound

Ultrasound imaging is achieved through high-frequency (1–10 MHz) sound irradiation. The intensity of the reflected soundwave is dependent on the properties of the materials involved. Ultrasound images are normally collected without a reference point, so these images are difficult to correlate in three-dimensions, which substantially limits the applicability of ultrasound imaging in the context of computational haemodynamics. Like PC-MRI, Doppler ultrasound can be used to provide real-time measurements of the blood flow velocities within the vessel.

1.4 Blood flow modelling

Blood flow modelling entails handling complex phenomena which are characterised by different time and space scales. Approximations are mandatory in order to emulate the behaviour of macroscopic sections of the vascular tree within a reasonable computational time, and therefore to give useful insights into the interplay effects between blood flow patterns and mechanisms associated with vascular pathologies.

In contrast to invasive medical imaging techniques, the simulation approach permits one to study any possible geometry with no danger to the patient [3, 16, 20]. Furthermore, blood flow modelling provides the flow field description in terms of

local velocity, pressure and stress, at any point with a good accuracy [20].

Finally, blood flow modelling enables one to study geometries which do not exist in reality; consequently, the clinician can predict the result of treatment procedures thereby aiding his/her expertise with a more rigorous and precise approach for pre-operative treatment planning optimisation purposes [20].

In the next section, we review several works which stress the importance of numerical techniques in the understanding of blood flow circulation processes, motivate their employment to overcome the limitations connected to *in vivo* or *in vitro* measurements, and help to identify the factors which lead to the initiation and progression of vessel malformations, and their rupture in the most extreme case.

1.4.1 Haemodynamic investigations of vascular diseases

In vivo measurements may be very dangerous for the patient. One approach to carry out risk-free measurements is to perform them on a liquid pumped into a reconstructed model, as approached in [21]; unfortunately, it is expensive and/or impractical to employ this strategy several times in a day [21] while efficient CFD applications can accurately model various vascular blood flows in hours with the employment of a sufficient computing power. Non-invasive imaging techniques, such as Magnetic Resonance angiography, can provide a fairly accurate velocity descriptions at a resolution of about 1 mm in space and 50 ms in time [19, 22–24]. CFD techniques offer the possibility to accurately examine blood flow in terms of velocity distribution, but can quantify several other parameters such as pressure, shear stress and vorticity, as well as at greater spatial and temporal resolutions by appropriately defining the velocity or pressure boundary conditions (blood flow velocity or pressure patterns on the borders of the vascular segment of interest); unfortunately, a typical clinical setting does not provide subject-specific boundary conditions. When these are not available and a scientist needs to approach a new vasculature he/she resorts to data available in the literature and usually employs prescribed pressure profiles at all boundaries or only at outflow ones (outlets) and velocity-controlled conditions for the inlets. Alternatively, one may assume that the outlet pressure is the outflow times the downstream peripheral resistance or that a structured tree is attached to the outlet (see [25] and references therein) in which the root impedance is estimated using a semianalytical approach based on a linearization of the equations which macroscopically describe the fluid flow behaviour in the continuum limit (Navier-Stokes equations, see Chapter 2).

A few studies [26–28] showed that there is no significant difference between the simulation results obtained through Newtonian and non-Newtonian blood flow models. In a Newtonian fluid, the relationship between the shear rate γ and the shear stress σ is linear:

$$\sigma = \mu\gamma \tag{1.1}$$

where the dynamic viscosity μ does not depend on the shear rate. The non-Newtonian behaviour of the blood is due to the proteins in the plasma, and the elasticity and aggregation of the red cells. This non-Newtonian behaviour is more evident when the shear rate is small such as it occurs in arterioles and capillaries [8]. The viscosity is affected by these properties and, in general, increases with shear rate and temperature (see [29] for the description of some blood flow models). The walls of the arteries are distensible and elastic, and their diameter varies with the transmural pressure. Their elastic behaviour is important for the propagation of the flow waves and the overall blood circulation.

Zhao *et al.* [30] demonstrated that the flow field behaviour simulated by employing fluid-structure interaction models is globally similar to the one achieved under the rigid wall assumption. However, they observed a general reduction in the magnitude of the WSS when the dynamics of the wall was considered. Unfortunately, fluid-structure interaction modelling entails assuming several subject-specific parameters since they are not usually provided in a standard clinical setting or by adopting state-of-the-art clinical equipment [30,31]. Boutsianis *et al.* [32] showed that a finite-volume fluid solver which is second order accurate in space can reproduce the *in vitro* blood flow behaviour of an anatomically reconstructed abdominal aorta aneurysm within an excellent approximation.

Several studies focussed on extra-cranial circulation further demonstrated the sensitivity of blood flow patterns to geometric details [33–36]. Through the modelling of either the blood via a CFD tool or the wall motion of femoral artery segments thanks to intravascular ultrasound (IVUS) imaging, Liu *et al.* [37] found that the changes in diameter and WSS were substantially influenced by local geometry displacements, enforcing the evidence that the coupling between fluid and wall motions may be important in atherosclerosis development. Myers *et al.* [33] obtained good agreement between WSS patterns carried out through a computer model of a right coronary artery and *in vitro* measurement.

In the study conducted by Hassan *et al.* [14] the blood flow simulation of an arterial aneurysm and a vein of Galen aneurysm contributed to understand important blood mechanisms and to assist the endovascular intervention process. Specifically, the origin of the malformed structure and abnormal blood flow field was attributed to the presence of a preaneurysmal stenotic segment between the two aneurysms, which caused a very high velocity. The success of the endovascular intervention was achieved through the application of twelve detachable coils and N-butyl cyanoacrylate injection which stopped the blood flow through the venous aneurysm.

Thanks to a three year study, conducted on a set of patients and aided by WSS

modelling via a CFD tool, Gibson *et al.* [38] showed that a low WSS promotes progression of atherosclerotic lesions. Their work and that conducted by Hoi *et al.* [39] underlines the importance of obtaining an accurate patient-specific spatial reconstruction, since a geometric change can provoke a blood flow variation up to an order of magnitude larger than the former.

Boyd *et al.* [40, 41] studied the effect of a simulated stenosis on the flow field around a bifurcation through two and three-dimensional lattice-Boltzmann models. They showed that the restriction is responsible for a highly rotational blood flow, low velocity and stress.

The CoW is fed by the left carotid and right carotid arteries, and the basilar artery. It allows the continuous supply of blood to the brain, even in the presence of flow disruptions upstream. How these vessels feed the brain is of interest, as haemodynamic perturbation at the CoW have consequences downstream in the cerebral vasculature.

Recently, Moore *et al.* [42] constructed a three-dimensional model to investigate the blood flow patterns of different anatomical models of the CoW. Simulations include blood flow modelling of the three most common configurations of the CoW: a complete CoW, a fetal P1 configuration and a missing A1 configuration.

Several studies have been carried out in the context of intra-cranial haemodynamics in the last fifteen years. Botnar *et al.* [43] obtained very good agreement between the velocity pattern in a carotid artery bifurcation achieved through CFD modelling and MR imaging of the *in vitro* specimen. Ujiie *et al.* [44] correlated the aneurysm hemodynamics to its aspect ratio, which is the ratio between its maximal diameter and that of the neck (area between the aneurysm and its host vessel), and concluded that an aneurysm with aspect ratio greater than 1.6 is prone to rupture.

Cebral *et al.* [3] made a remarkable effort to correlate haemodynamic factors to the risk of rupture, taking into account several aneurysm configurations. They conclude that a small inflow jet, small impingement region and a complex or unstable flow patterns are prone to yield high and oscillating stress, and cause rupture through elevated force and structural destabilisation at the walls.

Thanks to *in vivo* analysis of bifurcations of six canines, Meng *et al.* [45] showed that high WSS and high WSS gradient provoke aneurysm-type remodelling of the vessel wall.

Castro *et al.* [18] investigated the effects of unequal flow conditions in the ICAs on the haemodynamics of AcomA aneurysms. A significant difference in flow conditions between the ICAs, for instance that given by changing their relative phase, may cause the flow far from them to change rapidly. Since the geometry can have a large effect on the blood flow, the boundary conditions were set as far as possible from the aneurysms or other complex parts, so that a Womersley flow profile [46] at the inlet would be more accurate [14, 47]. In general, when blood impinges on an aneurysm wall, there is

a stagnation point at which the pressure is maximal but the WSS is zero. The WSS is very high around it, caused by the bloodstream turning along the wall [14,47]. The points of rupture have a relatively high pressure and WSS.

Castro *et al.* [18] found that curvatures in the parent artery upstream to the aneurysm neck significantly influences the direction of the inflow jet. The WSS becomes lower and shifted towards the neck if the inlets are positioned so as to have shorter parent arteries. Milner *et al.* [48] demonstrated how averaged flow rate wave forms or idealized carotid bifurcation models can substantially mask interesting haemodynamic features that may be significant in the characterisation of vascular pathologies. Venugopal *et al.* [49] remarked that it is important to impose patient-specific inlet flow rates to accurately simulate the overall haemodynamics.

Alastruey *et al.* [13] studied the effects of anatomical variations on the cerebral outflows of the CoW, both in healthy conditions and after a complete occlusion of an ICA or VA. When one of the A1ACAs and P1PCAs is absent, the flow rate through the communicating arteries is higher. The flows in the efferent arteries changed by less than 15%. This suggests that flows through the communicating arteries are sufficient to supply all areas of the brain in subjects with a small A1 ACA or P1 PCA. In the anatomies tested with missing communicating arteries, the flows in the other arteries changed by less than 1 %. The occlusion of the VA has relatively little effect compared to the occlusion of the ICA. They also showed that the AcomA is a more critical collateral pathway than both PcomAs if an ICA is occluded.

The factors which lead to vessel rupture are not fully understood yet. As shown later, several works have been carried out to find those factors. Recently, Chatziprodromou *et al.* [50] proposed a metric to assess the risk of cerebral aneurysmal growth and atherosclerosis formation assuming a combination of parameters are relevant in that assessment. Specifically, their risk metric is a function of the WSS, its temporal variation and the vorticity of the flow; this choice is associated with the assumptions that (a) a substantial chaotic blood flow provokes vasodilation and thus aneurysm formation [51] and expansion, and (b) a low and oscillatory WSS causes vessel wall thickening [38,52]. The risk metric was calculated by measuring these variables under normal simulation conditions and under either: (i) increased blood flow conditions, (ii) increased pulsatility or (iii) a combination of the two. Simulations with a cerebral fusiform aneurysm highlighted the region which may exhibit further growth as at the dome of the aneurysm, which likely correlated with the location of aneurysm rupture. Unfortunately, the risk factor is a function of parameters that are not yet experimentally measured; they should be used to properly formulate the dependence of the risk factor on its variables, like local WSS and vorticity. Second, an important factor which determines aneurysm formation may be a reduced vascular tone, which is not directly caused by high WSS but is a function of multiple factors not well known yet.

Finally, Chatziprodromou *et al.* correlated aneurysm and atherosclerosis formations only to the substantial change in blood flow dynamics that can occur, for instance, under stress or exercise conditions, but does not evaluate the vascular pathology risk within a subject-specific blood flow condition.

Works by Shojima *et al.* [53], Joua *et al.* [54] and Boussel *et al.* [55] provided evidence of the correlation between aneurysm growth and low WSS which contradicts the conclusion reported by Hassan *et al.* [47]. Specifically, it was shown that, while high WSS induces vasodilation initiation, low WSS facilitates the growing phase due to acute wall remodelling and degeneration which, ultimately, may bring to rupture. Nonetheless, aneurysm aspect ratio and spatially averaged WSS may be positively correlated with risk of rupture [53]. Specifically, this point is not in conflict with the conclusion reported in [53–55] since high and low WSS are likely to be mixed in the same aneurysm area.

The @neurIST project [56] had the objective to provide an integrated decision support system to assess the risk of aneurysm rupture and to optimise their treatments. Specifically, the @neurIST project intends to provide an IT infrastructure for the management, integration and processing of data associated with the diagnosis and treatment of cerebral aneurysm and subarachnoid haemorrhage. We are part of the GENIUS (Grid Enabled Neurosurgical Imaging Using Simulation) project (see Chapter 5). It aims at achieving some goals of the @neurIST project. However, the GENIUS project stresses the aspects associated with the integration of the software applications involved and high performance computing (HPC); this allows simulation turnaround and user times to be shortened, such that patient-specific medical simulation results can be obtained in a time frame which is clinically relevant.

1.5 Blood flow modelling as a pre-operational treatment planning tool

It is currently difficult to assess whether the treatment will be successful before it is carried out. Consequently, cerebral blood flow simulations may be an essential tool for clinicians to assess the effects of treatments before they are administered.

It is essential to ensure that after a treatment there is a sufficient blood supply to all areas of the brain, that blood flow through aneurysms or AVMs and inferred to be excessive is reduced and that haemodynamic factors, such as the WSS and blood pressure, are favourable and do not lead to complications. Blood flow simulation is a valuable approach to confirm assumptions and better understand haemodynamic factors. For instance, it is commonly believed that complex flow patterns are prone to develop atherosclerosis and vasodilation; unfortunately, several related mechanisms and their importance are not well understood yet. Therefore, with the use of CFD

techniques ineffective or unnecessary treatments may be potentially avoided, while risky patient-specific conditions may be treated with more precision.

Most of the effort of this PhD course has been directed to construct a modelling pipeline which is easy to use, enables the effective employment of distributed computing and large computational resources, and is powerful in terms of visual results, interactive capabilities and turnaround times. The simulation technique is the lattice-Boltzmann method (LBM) which is the topic of Chapter 2; the other aspects will be discussed in detail in the other chapters. We anticipate that even though the accuracy of the segmentation techniques and lattice-Boltzmann models employed in our work are not very high, we are seeking more accurate approaches; furthermore, we have successfully addressed several difficulties connected to our modelling pipeline.

1.6 Summary

This chapter has provided an overview about haemodynamics. Specifically, it presented a basic information on blood circulation. Then, it focussed the attention on the description of cerebrovascular diseases and on medical imaging techniques. We have also reviewed several studies conducted in the last two decades to increase the knowledge of several aspects regarding neurovascular pathologies, their formation, progression and haemodynamic features. In this context, CFD techniques have played a crucial role. Finally, this chapter ends by discussing the potential employment of the simulation approach based on the lattice-Boltzmann method as a means to aid pre-operative treatment planning.

Chapter 2

The lattice–Boltzmann method

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality

Albert Einstein

In this chapter a survey of computational fluid dynamics (CFD) techniques including in particular the lattice-Boltzmann method (LBM) is provided. Specifically, basic concepts and limitations of the main CFD approaches are outlined. Then, the fundamental theory and work-flow of the LBM are given in detail and finally, some variations of the most widely used LB schemes are discussed with some emphasis on their advantages and limitations.

2.1 Computational Fluid Dynamics

Computational fluid dynamics (CFD) is the conventional methodology for fluid flow modelling. Basically, CFD techniques directly solve the Navier-Stokes (NS) equations through explicit discretisation. For instance, the finite-difference method [57] replaces the partial derivatives of the NS equations with finite difference discretisations derived from truncated Taylor series expansions, usually first or second order accurate. Then, starting from an initial condition of the flow field and handling boundaries with some care, an iterative approach based on the aforementioned discretisations provides the flow field at each time step.

The NS equations are valid if macroscopic length and time scales, usually estimated as the characteristic size of the system considered and the time with which macroscopic field variations take place, are much larger than the molecular ones, *i.e.* the mean free path and time between subsequent particle collisions. In this limit the fluid is always in local thermodynamic equilibrium, and considered as a continuum.

Within this approximation, which holds for a large number of phenomena, the NS equations yield an accurate representation of the macroscopic fluid behaviour.

The main NS equation-based CFD approaches are the finite-difference [57], finite-volume [57] and finite-element [58] methods. Unfortunately, they fail to describe processes characterized by microscopic time and space scales. Conversely, the huge number of atoms in a macroscopic region prevents the employment of molecular- or atomistic-level models, and consequently their adoption is limited to the study of very small systems for a time of the order of the microsecond or less.

Mesoscopic formulations are well-suited to treat *complex fluids*, such as multi-component and multiphase fluids, and also multiscale systems by embedding microscopic models *e.g.* the molecular dynamics (MD) method. The integration of mesoscopic and microscopic approaches is the only viable strategy to study a system for which fine details, such as those located close to interfaces of a fluid mixture, play a central role in some parts of it. Thus, these models can be applied to describe small scale phenomena, while macroscopic techniques can quickly characterise the rest in order to keep the total computational time reasonable. The paper of Li and Liu [59] (and references therein) offers a vast survey of microscopic approaches, classical and *ab initio* MD methods, as well as mesoscopic ones *e.g.* the smoothed particle hydrodynamics and mesh-free Galerkin techniques.

Other important approaches in the context of mesoscopic modelling are the dissipative particle dynamics method [60] and the lattice-gas technique proposed by Malvanets and Kapral [61]. The former is a mesh-free approach based on the dynamics of fictitious mesoscopic particles which basically model large clusters of microscopic fluid particles; the latter describes the motion of mesoscopic particles placed at the sites of a rectilinear lattice through the application of stochastic rules.

2.2 The lattice-Boltzmann method

Nowadays, the LBM has substantial importance in various research fields which deal with mesoscopic and continuum techniques [62]. First, a literature survey associated with the LBM and its applications is presented. Here, an emphasis on haemodynamics is given; then, the attention is focussed on the theoretical aspects.

2.2.1 Overview

The lattice-Boltzmann method is an effective and accurate simulation tool for the investigation of several different problems [62, 63]. The accuracy and performance of the LBM have been compared to those of the finite-difference [64, 65]), finite-volume [66–69] and finite-element methods [69–72]. These studies reveal that the LBM can provide simulation results more quickly than NS-based CFD approaches if

a prescribed accuracy is aimed for. The LBM is simple, explicit in time and local in space. Therefore, it yields very efficient executions on parallel machines [73]. The LBM does not guarantee incompressibility but, in contrast to CFD techniques, it does not require a computationally intensive elliptic Poisson equation to be solved at each iteration to obtain the pressure field. These are important aspects for which one may prefer the LBM over other CFD techniques for the simulation of large systems.

The accuracy of the LBM in haemodynamics has been demonstrated [68,74] and, in this context, the method has been applied to several different studies [41, 68, 75–82]. In haemodynamics a high level of parallelism is essential to rapidly capture the complexity of a typical problem. Furthermore, the stress tensor can be directly calculated from the non-equilibrium components of the distribution functions [83]; NS-based CFD approaches, instead, require the use of interpolation techniques which increase the programming complexity, the computational time and inaccuracy.

The LBM has been employed for a large variety of purposes. It is also well-suited to simulate turbulence, non-ideal gases, multicomponent fluids, fluids with suspensions or in porous media, chemical-reactive flows, magnetohydrodynamical systems, microchannel and non-Newtonian and anisotropic fluids. Finally, numerous studies in turbulence modelling have been carried out through lattice-Boltzmann models. Succi [62], Chen and Doolen [63] provide an extensive literature review of all these research fields.

The success of the LBM method in the simulation of nonideal gases and multicomponent fluids with or without the presence of porous media, multi-phase and microchannel flows resides in its kinetic nature and thus the ability to incorporate small length-scale properties. Explicit interface tracking is not needed; this permits one to effectively simulate fluid flows with complicated interface structures confined in complex geometries due to simple and efficient boundary condition methods.

Fluid dynamics by means of the LBM with moving boundaries has already been investigated by Hoekstra *et al.* [84] who exploited the boundary condition method in [78]. However, Hoekstra *et al.* considered the walls as massless zero-thickness structures and the equations of motion associated with the walls were not integrated which results in an unrealistic physical representation.

Under the assumption that we have a Newtonian isothermal fluid with constant viscosity and density, the equations of conservation of mass and momentum that describe the macroscopic flow behaviour become the Navier-Stokes equations for a Newtonian, incompressible and isothermal fluid:

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla P + \eta \nabla^2 \mathbf{u} + \mathbf{G}, \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2.2)$$

where \mathbf{u} , P , ρ and ν are the fluid velocity, pressure, density and kinematic viscosity ($\nu = \eta/\rho$, η is the dynamic viscosity) respectively, and \mathbf{G} is the body force. The first equation describes momentum conservation and asserts that the acceleration is equal to the sum of the pressure, viscous and external force while the second one describes the mass conservation.

2.2.2 The lattice-Boltzmann method and the Lattice Gas Automata technique

The origins of the LBM reside in the development of the Lattice Gas Automata (LGA) technique [85–87] and in the study of lattice gases [87–90]. In the LGA method, a set of discrete particles is constrained to lie on the sites of a regular lattice and to have discrete velocity vectors corresponding to the nearest neighbours of each node. At each time step, the dynamics advance by moving each particle to one of its nearest lattice sites (advection) whilst ensuring momentum conservation during the collision stage. The method is very simple to implement, fast and unconditionally stable but suffers from some drawbacks, such as lack of Galilean invariance¹, and significant noise in the results due to integer calculations and fluctuations inherently associated with the LGA technique. These limitations do not afflict the LBM and the approaches based on the direct discretisation of the NS equations.

2.2.3 The lattice-Boltzmann method and the Boltzmann kinetic theory of fluids

Fluids may be investigated through a kinetic theory developed by Boltzmann [91], a *bottom-up* methodology that allows one to directly obtain macroscopic and thermodynamics descriptions from their microscopic ones. It has been shown that the LB equation is a special finite difference form of the continuous Boltzmann equation with some approximations applied for hydrodynamic simulation purposes [92–95]. Some aspects of the LBM, *e.g.* isotropy and Galilean invariance, have been studied by Lallemand *et al.* [96]. Analytical solutions of simple systems with the LBM [97] has further improved its understanding.

Transport phenomena can be studied through kinetic approaches based on the Boltzmann equation, derived from Newton’s laws of motion in the limit of large numbers of particles. The Boltzmann transport equation for monoatomic gases asserts that the change of number of molecules nf , where n is the local number density, $f = f(\mathbf{x}, \vec{\xi}, t)$ is the distribution function and \mathbf{x} and $\vec{\xi}$ are the position and velocity

¹A frame of motion is Galilean invariant if the equations of motion do not change in all other frames; Galilean variance leads to a velocity-dependent pressure and a density-dependent term in the macroscopic description.

vectors of a molecule respectively, is equal to the sum of the convection term, the contributions due to the body force \mathbf{G} and the collisions:

$$\frac{\partial f}{\partial t} = -\vec{\xi} \cdot \frac{\partial f}{\partial \mathbf{x}} - \mathbf{G} \cdot \frac{\partial f}{\partial \vec{\xi}} + Q(f), \quad (2.3)$$

where $Q(f)$ is the quadratic collision operator. This equation can accurately describe the fluids in the *kinetic regime*, *i.e.* the molecules can be considered *point-wise, structureless* particles such that the contribution associated with rotational and vibrational degrees of freedom do not play a significant role at the macroscopic level, and the interaction range is negligible with respect to the free-collision motion scale of the particles. In other words, the last equation can be employed for fluids with very simple molecules whose sphere of influence is much smaller than the average volume per molecule. Under these circumstances, f is close to the Maxwellian distribution

$$f^{(eq)}(\vec{\xi}) = \rho(2\pi RT)^{-D/2} \exp(-(\mathbf{v} - \vec{\xi})^2/2RT), \quad (2.4)$$

where R , T , D and \mathbf{v} are the ideal gas constant, the temperature of the fluid, the spatial dimension and the macroscopic velocity respectively. Both Navier-Stokes and Euler equations can be derived from the Boltzmann equation using the Chapman-Enskog procedure.

2.2.4 Details of the lattice-Boltzmann method

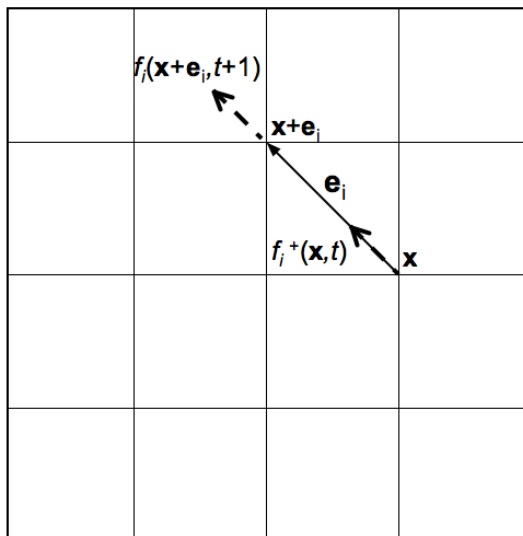


Figure 2.1: Schematic illustration of the advection step of the lattice-Boltzmann method (see text for details on the symbols).

McNamara and Zanetti [88] proposed that the Boltzmann equation should be solved directly by modelling the set of particles in the LGA method as single-particle

distribution functions instead of finding them through computationally-intensive averaging of occupation numbers:

$$f_i(\mathbf{x} + \mathbf{e}_i, t + 1) = f_i^+(\mathbf{x}, t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t), \quad (2.5)$$

where $f_i(\mathbf{x}, t)$ is the average number of particles at the lattice position \mathbf{x} and time t moving at velocity \mathbf{e}_i , the superscript “+” denotes a post-collisional state and $\Omega_i(\mathbf{x}, t)$ stands for the lattice version of the collision integral in the continuum Boltzmann equation. The advection stage is represented in Fig. 2.1 for a two-dimensional computational lattice of 5×5 lattice sites. Various models have been proposed in order to represent the collision operator [89,98]; the most widely adopted approximation of the continuum collision term is that suggested by Bhatnagar, Gross and Krook [99] for which the local distribution function is assumed to evolve towards its equilibrium value, at a rate controlled by a single relaxation parameter τ :

$$\Omega \approx \frac{f^{(eq)}(\mathbf{x}, t) - f(\mathbf{x}, t)}{\tau}. \quad (2.6)$$

In light of this, Chen *et al.* [100] and Qian *et al.* [101] proposed the *lattice BGK equation*:

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)}{\tau}, \quad (2.7)$$

with various two and three-dimensional lattice structures and explicit values of the coefficients within the formula of the equilibrium distribution functions

$$f_i^{(eq)} = \rho w_i \left(1 + \frac{\mathbf{e}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right), \quad (2.8)$$

where $f_i^{(eq)}$ retains the terms of the Taylor series expansion of the Maxwellian distribution up to $O(Ma^2) = O((\mathbf{u}/c_s)^2)$, w_i is a weight coefficient and c_s is the speed of sound; the hydrodynamical density and velocity ρ and \mathbf{u} respectively are determined in terms of the distribution functions:

$$\rho = \sum_i f_i = \sum_i f_i^{(eq)}, \quad \rho \mathbf{u} = \sum_i \mathbf{e}_i f_i = \sum_i \mathbf{e}_i f_i^{(eq)}. \quad (2.9)$$

The discrete velocities \mathbf{e}_i and the weight coefficients w_i must be chosen in order to assure spatial isotropy. As discussed in [62] and references therein, the isotropy conditions for a lattice are the following:

$$\sum_i w_i = 1, \quad (2.10)$$

$$\sum_i w_i e_{i\alpha} e_{i\beta} \propto \delta_{\alpha\beta}, \quad (2.11)$$

$$\sum_i w_i e_{i\alpha} e_{i\beta} e_{i\gamma} e_{i\delta} \propto (\delta_{\alpha\beta} \delta_{\gamma\delta} + \delta_{\alpha\gamma} \delta_{\beta\delta} + \delta_{\alpha\delta} \delta_{\beta\gamma}). \quad (2.12)$$

where δ_{ij} is the Kronecker delta function. The lattice BGK (LBGK) models are isotropic, Galilean invariant, conserve mass and momentum during local particle collision and do not exhibit noise caused by integer calculations. Furthermore, they are second-order accurate in time and space (see below). However, in contrast to the LGA method, the LBM is not unconditionally stable.

The LBM may be regarded as a specific finite-difference discretisation of the Boltzmann equation [95]. In fact, adopting the single time relaxation approximation for the collision term and assuming no external force, the evolution of the distribution function f_i becomes

$$\frac{\partial f_i}{\partial t} + \mathbf{e}_i \cdot \nabla f_i = -\frac{f_i - f_i^{(eq)}}{\tau}. \quad (2.13)$$

If the time derivative is approximated by a first order time difference and the convective term by a first order space discretisation, the finite difference equation for f_i is

$$\begin{aligned} \frac{f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t + \Delta t) - f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t)}{\Delta t} + \\ \frac{f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t) - f_i(\mathbf{x}, t)}{\Delta x} = -\frac{f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)}{\tau}, \end{aligned} \quad (2.14)$$

which is the standard LBGK equation with $\Delta t = \Delta x = \Delta y = 1$:

$$f_i(\mathbf{x} + \mathbf{e}_i, t + 1) - f_i(\mathbf{x}, t) = -\frac{f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)}{\tau}. \quad (2.15)$$

At a first glance, the above discretisation resembles a first-order accurate discretisation in space and time, but the above equation is second-order accurate in space and time [102]. In fact, by applying the trapezoidal rule to the integral of the collision term we have

$$\begin{aligned} \frac{f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - f_i(\mathbf{x}, t)}{\Delta t} = & -\frac{f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) - f_i^{(eq)}(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t)}{2\tau} \\ & -\frac{f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)}{2\tau} + O(\Delta t^2), \end{aligned} \quad (2.16)$$

which becomes explicit and specifically the LBJK equation reported above by setting $\Delta t = 1$ and by performing the following variable changes for the distribution function and τ :

$$\tau \leftarrow \tau + \frac{1}{2} \quad (2.17)$$

$$f_i(\mathbf{x}, t) \leftarrow f_i(\mathbf{x}, t) + \frac{f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)}{2\tau} \quad (2.18)$$

The stress tensor, $\sigma_{\alpha\beta}$, can be recovered from the non-equilibrium components of the the distribution function $f_i^{(neq)} = f_i - f_i^{(eq)}$ [83]:

$$\sigma_{\alpha\beta} = -p\delta_{\alpha\beta} - \left(1 - \frac{1}{2\tau}\right) \sum_{i=0} f_i^{(neq)} e_{i\alpha} e_{i\beta}. \quad (2.19)$$

The von Mises stress is often employed in haemodynamic studies to evaluate the effective stress [103]

$$\sigma_{\text{eff}} = \sqrt{\frac{A + 6B}{2}}, \quad (2.20)$$

where A and B are defined in Equations (2.21) and (2.22):

$$A = (\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 \quad (2.21)$$

$$B = \sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2. \quad (2.22)$$

Finally, the LBGK equation is valid if

- (a) any two particles in the system are uncorrelated before and after their brief collision; thus, the *molecular chaos condition* is satisfied²;
- (b) the lattice spacing is much bigger than the mean free path; thus, we achieve the *low Knudsen number limit*³;
- (c) the velocity is much smaller than the speed of sound in the system considered; consequently, we are in the *limit of low Mach number*.

Conditions (b) and (c) are necessary since the LB distribution functions are reliable approximations in the limit of low Knudsen and Mach number, as described below.

2.2.5 Derivation of the Navier-Stokes equations

In this sub-section, it is shown that the LB equations recovers the Navier-Stokes ones in the limit of low Knudsen and Mach numbers via the Chapman-Enskog procedure. Specifically, LB models approach the Navier-Stokes equations with error terms, referred to as compressibility errors, proportional to the Knudsen number squared and Mach number squared or cubed [105]. The Knudsen number Kn is [106]:

$$Kn = \frac{l}{h} = \frac{\lambda c^*}{h}, \quad (2.23)$$

where l is the mean free path, h is the characteristic length of the channel, c^* is the average molecular velocity due to thermal excitation and $\lambda = l/h$ is the relaxation

²The many-body correlation function becomes a product of single particle distribution functions.

³See the paper by Toschi and Succi [104] for a deep theoretical discussion and a model well-suited for simulations affected by high Knudsen numbers.

time. Toschi and Succi [104] remarked that the accuracy of the LB model depends on the Knudsen number, Mach number and lattice resolution. However, Maier *et al.* [107] pointed out that, while for a duct flow problem neither the Mach number nor the Knudsen separately represent good indices of the errors, their product yields a fair estimate of the error behaviour.

The Chapman-Enskog expansion is employed to derive the macroscopic hydrodynamic equations. Employing the Chapman-Enskog expansion, which is valid under assumptions (b) and (c), a Taylor series expansion in space and time, and accurate to second order in the Knudsen number $Kn = \epsilon$ is applied to the evolution of f_i :

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)}{\tau}, \quad (2.24)$$

where Δx and Δt are of the same order of ϵ ; hence,

$$\frac{\partial f_i}{\partial t} + \mathbf{e}_i \cdot \nabla f_i + \frac{\epsilon}{2} \left[(\mathbf{e}_i \cdot \nabla)^2 f_i + 2\mathbf{e}_i \cdot \nabla \frac{\partial f_i}{\partial t} + \frac{\partial^2 f_i}{\partial t^2} \right] = -\frac{f_i - f_i^{(eq)}}{\epsilon \tau}. \quad (2.25)$$

Assertions (b) and (c) permit the application of the Chapman-Enskog expansion and thus

$$\partial_t = \epsilon \partial_{t_0} + \epsilon^2 \partial_{t_1}, \quad \partial_x = \epsilon \partial_x, \quad (2.26)$$

where t_0 and t_1 denotes the time-scales associated with advection (or convective effects) and diffusion (or viscous effects) respectively. The single-particle distribution function can be decomposed into its equilibrium part and non-equilibrium one, *i.e.* into terms of different orders of magnitude:

$$f_i = f_i^{(eq)} + \epsilon f_i^{(neq)}, \quad (2.27)$$

with $f_i^{(eq)} = f_i^0$ and $f_i^{(neq)} = f_i^1 + \epsilon f_i^2 + O(\epsilon^2)$. The converved quantities lead to constraints on high-order terms $f_i^{(j)}$ (solvability conditions):

$$\sum_i f_i^j = 0, \quad \sum_i \mathbf{e}_i f_i^j = 0, \quad j > 0; \quad (2.28)$$

thus the higher order terms f_i^1, f_i^2, \dots do not contribute to the macroscopic density and momentum, *i.e.* local mass and momentum are conserved during collision. By employing Eq. 2.25 and by regrouping terms in power orders of ϵ we obtain

$$\frac{\partial f_i^{(eq)}}{\partial t_0} + \mathbf{e}_i \cdot \nabla f_i^{(eq)} = -\frac{f_i^1}{\tau}, \quad (2.29)$$

to zero order in ϵ , and

$$\begin{aligned} \frac{\partial f_i^{(eq)}}{\partial t_0} + \frac{\partial f_i^1}{\partial t_1} + \mathbf{e}_i \cdot \nabla f_i^1 + \frac{\partial^2 f_i^{(eq)}}{\partial t_0^2} + 2\mathbf{e}_i \cdot \nabla \frac{\partial f_i^{(eq)}}{\partial t_0} + (\mathbf{e}_i \cdot \nabla)^2 f_i^{(eq)} = \\ \frac{\partial f_i^1}{\partial t_1} + \left(1 - \frac{2}{\tau}\right) \left[\frac{\partial f_i^1}{\partial t_0} + \mathbf{e}_i \cdot \nabla f_i^1 \right] = -\frac{f_i^2}{\tau} \end{aligned} \quad (2.30)$$

to first order in ϵ . Using the last two equations, summing over all the directions i , and exploiting the solvability conditions we recover the following mass and momentum equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \quad (2.31)$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\nabla \cdot \Pi, \quad (2.32)$$

where the momentum flux tensor has the form

$$\Pi_{\alpha\beta} = \sum_i (\mathbf{e}_i)_\alpha (\mathbf{e}_i)_\beta \left[f_i^{(eq)} + \left(1 - \frac{1}{2\tau}\right) f_i^1 \right], \quad (2.33)$$

The detailed form of the momentum flux tensor depends on the lattice structure of the current LB model. For the LB D2Q9 (two dimensions and nine velocities) model,

$$\Pi_{\alpha\beta} = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta + \nu(\nabla_\alpha(\rho u_\beta) + \nabla_\beta(\rho u_\alpha)), \quad (2.34)$$

where $p = \rho/3$ is the pressure and $\nu = (2\tau - 1)/6$. With this form of momentum flux tensor, the momentum equation becomes the Navier-Stokes one with no external force:

$$\partial_t(\rho u_\alpha) = -\partial_\beta(\rho u_\alpha u_\beta) - \partial_\alpha p + \mu \partial_\beta \partial_\beta u_\alpha. \quad (2.35)$$

The continuity and momentum equations reduce to the incompressible ones if the density is constant.

2.2.6 Kernel of the lattice-Boltzmann method

In Fig. 2.2 we depict a pseudo-code of the LB computational work-flow based on the simple LBGK model. Typically, in a preprocessing stage two buffers are allocated to store the distribution functions at every lattice site belonging to the computational domain Ω and for any lattice direction i . These buffers are usually coined “source” (f_{source}) and “destination” (f_{dest}) buffers to recall that the streaming stage of the LBM fetches the data at a lattice site and stores them at nearby ones. At the beginning of the simulation, every element of f_{source} is usually set equal to the corresponding equilibrium distribution function calculated with a unitary density and zero velocity (steps a_P and b_P).

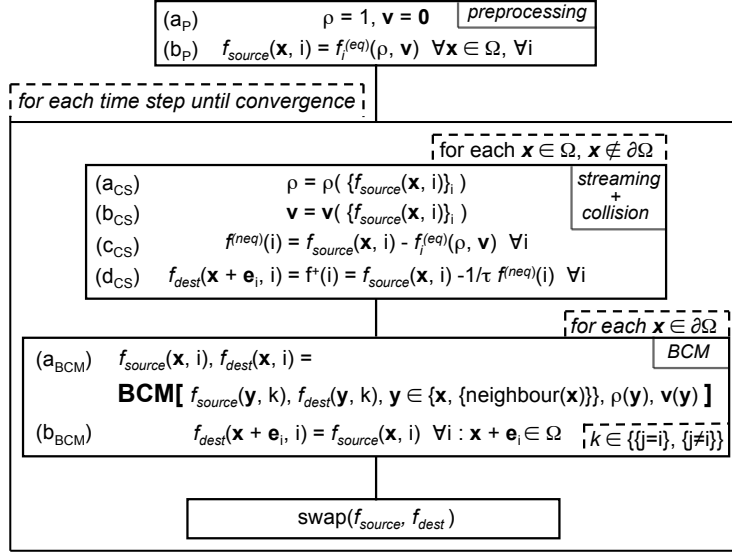


Figure 2.2: Schematic illustration of the computational work-flow of a LBGK method (see text for details on the symbols).

Then, the LB simulation core proceeds until a user-defined number of time steps and/or numerical convergence are reached by repetitively loading the elements of f_{source} and storing them in f_{dest} by following the standard collisional and propagation rules for the interior lattice sites (namely *streaming + collision* stage “CS”)⁴. A specialised function (in the figure “BCM” which stands for “Boundary Bondition Bethod”), instead, is applied to the boundary lattice sites where the elements of f_{dest} –and eventually those of f_{source} – are determined on the basis of the data of the current site \mathbf{x} and, in general, the neighbouring ones as well. At the end of each time step, f_{source} and f_{dest} are swapped.

In particular, during stage CS the local density and velocity are computed through equations 2.9 (steps a_{CS} and b_{CS}), the non-equilibrium distribution functions through step c_{CS} while the streaming and collision steps are merged in step d_{CS} where $f^+(i)$ indicates the local post-collisional i -direction-wise distribution function. During stage BCM, instead, for some directions (in the figure generically denoted by the letter “i”) the elements of f_{source} and f_{dest} are computed on the basis of the current and neighbouring lattice sites, same and/or different lattice directions (in the figure $k = i$ or not) and the boundary data (“y” can be located at the boundary). Normal data streaming occurs towards any lattice site within Ω .

We discriminate BCMs that need data stored at one or more neighbouring sites, namely “non-local”, from the others (“local”)⁵. It is worth noting that local BCMs

⁴An interior lattice site is a site for which all the lattice velocity vectors departing from it are not crossed by the domain boundary $\partial\Omega$.

⁵Note that the parameters related to a point on the boundary surface (like the components of the

are particularly well-suited for parallel approaches since they do not require one to implement specialised processor-communication patterns to handle neighbouring information exchange. Furthermore, non-local BCMs can create significant computational and communicational unbalances because the processors may have very different numbers of boundary sites and thereby may be subjected to different workloads and amounts of data communications. In Chapters 3 and 6, we will describe and test various local and non-local BCMs.

2.3 Non-standard LB schemes and improvements

Several works have aimed at augmenting the numerical properties of the LBM, like its stability region, its accuracy and its convergence rate.

Some incompressible LB models eliminate the errors dependent on the Mach number if the fluid flow is steady [105,108]. Other LB models improve the stability region and the accuracy in the presence of large pressure gradients. In particular, a LB simulation becomes unstable if the equilibrium distribution function becomes negative; this tend to happen when the magnitude of the local velocity exceeds ≈ 0.3 in lattice units which in turn may be caused by large pressure gradients. The lattice BGK scheme with general propagation presented by Guo *et al.* [109] increases stability but can yield a certain numerical diffusion. This scheme has been coupled with a nonideal model [110] which takes the particle size into account in order to provide a method in which the compressibility errors are well controlled in the presence of large pressure gradients.

Assuming that a physical phenomenon takes a total time T , the number of iterations required to reach the fully developed solution is $T/\Delta t$. In the LB model, the time step and lattice size depend either on the viscosity or the Mach number Ma :

$$\Delta t = \frac{1}{\sqrt{3}} \Delta x \frac{Ma}{|\mathbf{u}_{max}|}, \quad (2.36)$$

$$\Delta t = \frac{(2\tau - 1)/6}{\nu} (\Delta x)^2. \quad (2.37)$$

Therefore, the time step varies linearly with the lattice size for a fixed Mach number. Thus, poorly resolved computational grids are simulated very quickly if the Mach number is high. On the other hand, the second equation reveals that the time step varies as the square of the grid spacing for a fixed viscosity and relaxation parameter. This is the same situation encountered in conventional NS equation-based fluid solvers explicit in time. Unfortunately, τ and thus the viscosity must be kept low if a high Reynolds number is required for a certain discretisation in space and time; this has the effect to decrease the time step in physical units $\Delta t^* = \frac{\tau-1/2}{3\nu^*} (\Delta x^*)^2$ (the symbol normal vector at that point close to a boundary site can be stored within the data of the latter.

* denotes physical unit) and the stability of the LBGK scheme [96]. An alternative strategy to increase Reynolds number is to decrease Δx maintaining the time step and relaxation parameter within stability range. To sum up, the LBM can be considered computationally expensive to simulate high Reynolds numbers flows.

A significant speedup in the convergence rate can be attained. Several LB methods have been developed [111–113] in order to speed up the simulation of stationary flows, especially in the low Reynold and Mach number regime. Kandhai *et al.* [70] proposed an iterative momentum relaxation (IMR) technique, where the applied body force is iteratively adapted on the basis of the change in fluid momentum which substantially accelerate numerical convergence. Mach number annealing can be applied in order to increase convergence rate of unsteady flows [103]. Unsteady simulations, instead, can be speeded up by appropriately varying the available parameters, *e.g.* pressure gradient and number of time steps, whilst maintaining constant the Reynolds number and the Womersley parameter (see [114] and Chapter 6).

The fractional volumetric scheme of Zhang *et al.* [115] reduces the particle effective evolution for each time step by a factor α , propagating only one fraction of the density to the neighbouring site and therefore can remove unphysical spurious invariants, increase stability, and is capable of simulating higher Reynolds number flows achieving smaller viscosities. On the other hand, the total number of time steps to reach convergence increases as $1/\alpha$.

In generalized LB equations and multiple-relaxation-time models [62,102,116] the various relaxation times may be adjusted in order to increase stability, damping non-hydrodynamic modes due to “ghost” variables [117] and decreasing spurious effects due to discretisations. Another successful result in this direction can be achieved enhancing the bulk viscosity [118].

Various approaches have been presented in order to tackle complex computational domains, either using grids that fit boundaries or adapting the computational meshes to the physical behaviour of the system. The Boltzmann equation is a first-order partial difference equation and thus the LBM can be easily adapted to non-uniform grids [119–121], body-fitted meshes [122,123] or grid refinement (multilevel) techniques [69,124–129]. Coarse-grained LB approaches can borrow the adaptive grid refinement concept [130,131] adopted in conventional CFD methods in order to increase or decrease the mesh resolution where required. Even multiscale LB approaches can yield faster results (within a prescribed accuracy) than finite-element and finite-volume techniques especially for time-dependent flows and high Mach numbers [69,128].

Finite-difference [132], finite-volume [133–135] and finite-element and Galerkin [136–138] schemes of the LBM are available. Finite-difference formulations can yield a significant increase in numerical viscosity. Finite-volume and finite-element

schemes permit to adapt the mesh to the physics and configuration of the system under consideration. Finite-element schemes are more accurate and stable than the others. For instance, the method in [138] is least-squares finite-element fourth-order accurate in space and second-order accurate in time.

Lattice kinetic schemes that recover the Navier-Stokes equations [139–141] include terms in the equilibrium distribution function that depend on the velocity gradient. These schemes (a) can be adapted to complex geometries and arbitrary meshes with the least-squares technique [141], and (b) do not need to store the distribution functions since they can be directly calculated from the computed velocity and density fields, thus drastically reducing memory requirement.

The fluid flow simulation through the LBM must face the problem of the adoption of suitable boundary condition methods, especially when the confining walls have a complex shape and are not aligned with the Cartesian planes. A discussion of boundary condition methods well-suited for generic geometries is presented in Chapter 3 where we also present new algorithms for pressure and velocity boundaries.

2.4 Computational aspects

The LBM differs from methods which are directly based on the Navier-Stokes equations in several algorithmic computational aspects. These features are related to the kinetic nature of the LBM:

1. the Boltzmann equation, from which the lattice-Boltzmann model is derived, consists of a set of first-order partial differential equations, instead of second-order as the Navier-Stokes ones, and the non-linear term of the Navier-Stokes equations is hidden in the quadratic velocity terms of the equilibrium distribution function, which is local;
2. small length-scale models can be incorporated more easily into LBM;
3. the spatial and velocity discretisations are coupled, which prevents the direct and easy realisation of models well-suited for problems involving different space scales. However, finite-element, finite-volume formulations, non-uniform and grid refinement formulations of the LBM overcome this obstacle;
4. the LBM can be inefficient for time-independent problems, especially when the lattice resolution is high and the Mach number is low. A low Mach number is a requirement for quasi-incompressible flows, including turbulence. However, the LBM remains at least competitive to old-fashioned CFD techniques in this area too. Furthermore, as discussed above, the convergence rate can be significantly improved either for steady or unsteady flows;

5. NS-based CFD techniques need to solve the Poisson equation for the pressure; conversely, the LBM computes the pressure through an equation of state. This feature makes the LBM capable to yield higher parallel performance than that achieved by employing several other approaches.

2.5 Summary

This chapter was focussed on the LBM and its applications with an emphasis on its advantages and disadvantages with respect to CFD techniques based on the direct discretisation of the Navier-Stokes equations. The theoretical description was provided in detail. In particular, the LBM is explicit, second-order accurate in space and time, and well-suited to solving problems involving turbulence, multi-phase, non-Newtonian fluid and/or haemodynamics. Then, studies regarding non-standard LB schemes aimed at improving stability, convergence and accuracy were reviewed. Finally, the main computational features of the LBM were listed. Specifically, its multi-level or non-Cartesian formulations can be employed to model multi-scale systems. Furthermore, the explicit and kinetic nature makes the LBM a competitive technique for the parallel simulation of time-varying fluid flows in complex geometries.

Chapter 3

The fluid solver, HemeLB

If your success is not on your own terms, if it looks good to the world but does not feel good in your heart, it is not success at all.

Anna Quindlen

In this chapter, we describe the parallel lattice-Boltzmann (LB) fluid solver utilised to perform the three-dimensional simulations carried out during this research. Its development has required a substantial time since it has been revised, optimized and benchmarked several times. Its performance has been investigated on various platforms; this required considerable attention, especially in the context of cross-site runs, which has often involved the employment of platforms and grid environments not yet mature for those tests and influenced by several issues.

The parallel fluid solver has been augmented by several algorithmic tricks which permit elimination of complex buffer accesses, computational branching and minimise data pattern irregularity and data locality such that the resulting computational core is very simple and fast. Furthermore, communication is minimised and the novel topology-aware domain decomposition technique is shown to be very effective, permitting tuning code execution in geographically distributed cross-site simulations.

The benchmark results presented indicate that very high performance can be achieved on a single processor core, a parallel machine and on geographically distributed platforms.

HemeLB is a part of a complete problem solving environment wherein the medical data from various imaging modalities are manipulated by the graphical editing tool described in Chapter 6 and rendered by the *in situ* parallel ray tracer presented in Chapter 4.

3.0.1 Overview

Fluid flow simulation of very large and complex systems requires the use of a suitable physical model, substantial computational resources, as well as applications capable

of exploiting them effectively. Much work has been done in order to speed-up single-processor and parallel lattice-Boltzmann simulations in regular systems. However, relatively few works are aimed at the improvement of lattice-Boltzmann simulations of fluid flow confined in complex geometries. The parallel LB fluid solver, coined HemeLB (“Heme” derives from Hemodynamics), represents a contribution in this direction. HemeLB is currently used to investigate cerebral blood flow behaviour in patient-specific systems (see Chapter 6).

In the context of LB simulations of fluid flows confined in non-regular systems, particularly porous media [142,143], a number of studies have been carried out using the strategy presented by Donath *et al.* [144] to reduce the memory requirements. This approach avoids the storage of data associated with obstacles: one-dimensional arrays, one for each data type, store the information about the fluid lattice sites which represent the volume of the fluid. The identification of the neighbouring distribution functions, needed during the LB advection stage, relies on an extra one-dimensional connectivity buffer (see discussion in Sec. 3.2). An approach which employs slabs to save memory is presented in detail by Argentini *et al.* [145] in the context of the implementation of the LBM for regular systems.

It is well known that the lattice-Boltzmann method is memory-intensive because the corresponding simulation requires several bytes per fluid site, little computation during collision and a substantial transfer of non-contiguous data during the advection stage. This is not ideal when using cache-based microprocessors for the simulation of a large system¹ because the arrays do not fit into the cache hierarchy² and the poor bandwidth and latency of the main memory cannot guarantee to keep the CPUs busy; as a result, it is difficult or impossible to achieve a high performance.

Pohl *et al.* [147] and Schulz *et al.* [148] presented two different “compressed grid” approaches which reduce the total memory consumption by a factor of almost two by exploiting the deterministic data dependencies which occur in the propagation step. The approach of Martys and Hagedorn [149] and Argentini *et al.* [145] is also useful to reduce the total memory consumption by reducing the source or the destination buffer (see Chapter 2 for this terminology) to three two-dimensional slab-like buffers only –instead of a large three-dimensional matrix. However, these memory-saving strategies do not solve the problem of modest performance on large systems since non-contiguous data needed during the propagation step often reside far from each other. One and three-dimensional loop blocking techniques [147,150,151] show a similar performance improvement with respect to the approach of Martys and Hagedorn [149] and Argentini *et al.* [145] by augmenting space locality in small systems. The technique which permits one to obtain a high performance on large systems seems

¹A system may be considered to be large if it does not fit in any of the caches.

²The interested reader should consult Kowarschik and Weiß [146] for more detailed discussions on cache memory organisation and effective utilisation.

to be the n -way blocking presented by Pohl *et al.* [147], Iglberger [151], and Velivelli and Bryden [152]. For three-dimensional systems, 4-way blocking relies on maximum exploitation of the first-neighbour LB kernel by combining a three-dimensional space blocking with a one-dimensional time one. Many time steps are performed on three-dimensional blocks of different shapes involving access to the whole domain at one time only. This strategy guarantees good spatial and temporal coherency, and delivers high performance (see also [153] and [154] for further benchmarks on cache-aware LB implementations). Relatively simple prefetch and preload techniques [155, 156] can lead to a significant increase in memory bandwidth and improve the overall performance substantially. However, it is difficult if not impossible to apply the 4-way blocking approach to flow simulation in “sparse” (non regular) systems because the data workflow is completely irregular.

Domain decomposition strategies based on cubes, slabs or parallelepipeds [73, 73, 157–159] yield good computational and communication load balancing when applied to regular systems; unfortunately, they are not well-suited to complex ones.

Some domain partitioning strategies for lattice-Boltzmann codes well-suited for non-Cartesian lattices already exist, and they are briefly reviewed here. The state-of-the-art of domain decomposition, represented by the multilevel k -way partitioning scheme [160, 161], can be used to partition non-regular systems. The need for users to explicitly implement these complex domain decomposition techniques can be avoided by using existing software such as the METIS library [162], which can be incorporated within parallel lattice-Boltzmann codes to obtain high quality partitions and thus good execution performance on a large number of processors. However, even state-of-the-art parallel multi-level implementations require $O(N/10^6)$ seconds on $O(N)$ graph vertices³ even when performed on $O(100)$ processors [160]. This leads to unreasonable elapsed times whilst tackling systems with $O(10^8)$ fluid lattice sites or more. Moreover, these partitioning algorithms require a substantial amount of memory, which makes the use of a parallel approach compulsory as well as many processors even for systems which are not very large. Generally speaking, these multilevel methods use recursive bisection schemes for the initial partitioning. The interfaces between partitions are then calculated and iteratively adjusted in order to improve workload distribution, which is very important during the parallel simulation. As a consequence, these strategies have a large associated computational cost; unfortunately, they do not guarantee optimal communication or computational load balancing. For instance, the parallel LB codes presented by Dupuis and Chopard [163] and Axner *et al.* [164] employ the METIS library [162] to partition the system. Moreover, large systems would require long pre-processing times due to the high computational cost associated with the multilevel k -way partitioning method adopted in the METIS

³In our case, a vertex in a graph corresponds to a fluid lattice site.

library [162].

The domain decomposition approach used in Pan *et al.* [165] is worthy of note. It combines a one-dimensional data representation with connectivity [144] and is well-suited for sparse systems, possessing an efficient parallelisation strategy based on the orthogonal recursive bisection (ORB) algorithm [166]. In the ORB approach, the computational grid is decomposed into two partitions such that the workload, due to fluid lattice sites only, is optimally balanced. The same process is then applied recursively $k - 1$ times to every partition, proceeding on the basis of orthogonal bisections. The total number of processors must be equal to 2^k . The amount of data required for an efficient implementation is proportional to the total number of fluid lattice sites N . Furthermore, the computational time scales as $O(N \log(N))$; while the domain decomposition is straightforward to implement and ensures good load balancing, it does not produce a satisfactory communication balance.

In the strategy employed by Wang *et al.* [5] whilst reading the input dataset where the disposition of the fluid lattice sites is stored the p_{id} -th N/P (P is the number of processors) fluid sites are assigned to the processor with rank p_{id} regardless of their location. The difference of fluid lattice sites per processor is either 0 or 1, depending on whether N/P is an integer or not. Thus, the resulting computational load balance is perfect. The recovery of the data to communicate is straightforward since the fluid lattice sites adjacent to neighbouring processor sub-domains, called here “interface-dependent lattice sites”, are stored in an orderly fashion [5]. Hence the communication pattern is regular and no data flow discontinuities occur. The method is simple, assures perfect load balancing, does not require CPU time to optimise the decomposition and demands a memory consumption of $O(N/P)$ per processor, since global data are not necessary. Wang *et al.* [5] maintained their approach to be superior to that of the ORB technique. The approach itself is not new; in fact, it has been exploited in the multilevel graph partitioning method as an initial processor-wise data distribution prior to the execution of the latter. However, such domain decomposition can be afflicted by poor communication balance even for simple systems, as will be shown in Sec. 3.3.1.

Finally, space-filling curves [167] have been used to decompose sparse geometries in computational fluid dynamics (CFD) applications [168]. The quality and the speed of these partitioning strategies is sub-optimal, while the computational cost usually increases as $O(N \log(N))$, making them expensive for very large systems.

3.1 The HemeLB model

The lattice-Boltzmann model adopted in the HemeLB code is the lattice Bhatnagar, Gross and Krook (BGK) D3Q15 (three-dimensional with 15 velocity directions) model

presented by Qian *et al.* [101]. For the tests reported in the publication of HemeLB and presented in [169] the D3Q15 model proposed by Zou *et al.* [105], coined D3Q15i, has been employed; that LB model is incompressible for a stationary fluid flow, but has a Galilean-invariant component, as discovered by my colleague Gary Doctors (University College London). However, the numerical results carried out with the D3Q15 and D3Q15i do not usually differ by more than 1%. The lattice BGK (LBGK) equation is

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{f(\mathbf{x}, t) - f^{(eq)}(\mathbf{x}, t)}{\tau} \quad (3.1)$$

where the local equilibrium distribution functions are

$$f_i^{(eq)} = w_i \left(\rho + \frac{\mathbf{e}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right), \quad (3.2)$$

where w_i is the weight coefficient, $c_s = \sqrt{1/3}$ is the speed of sound, \mathbf{e}_i is the velocity of the particle along the direction i , and the hydrodynamic density ρ and macroscopic velocity \mathbf{u} are determined in terms of the distribution functions from

$$\rho = \sum_i f_i = \sum_i f_i^{(eq)}, \quad \mathbf{u} = \frac{\sum_i \mathbf{e}_i f_i}{\rho} = \frac{\sum_i \mathbf{e}_i f_i^{(eq)}}{\rho}. \quad (3.3)$$

The discrete velocities \mathbf{e}_i and the weight coefficients w_i must be chosen in order to ensure isotropic hydrodynamics (see [105] for details) and are identical to those employed by Qian *et al.* [101].

3.1.1 Boundary condition methods

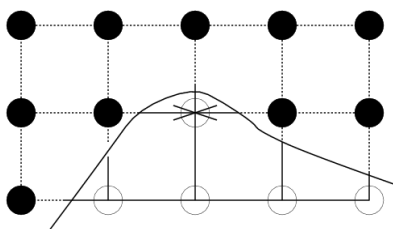


Figure 3.1: Schematic diagram of a smooth boundary for which both horizontal inward-pointing directions indicated by the arrows originate from solid sites. Lattice sites within the computational fluid domain are denoted by empty circles. Many boundary condition methods are unable to determine the unknown distribution functions at the central fluid lattice site.

The pressure and velocity boundary condition methods (BCMs) implemented in HemeLB are new. Here, we present them together with some preliminary concepts and one variant. Their development was motivated by the fact that, up to our knowledge, accurate and simple BCMs which can handle any boundary configuration such as that depicted in Fig. 3.1 are not available. Existing BCMs of a second-order accuracy for

the velocity field are very complex to implement and compute, and are not convincing (see below) thereby underpinning the need to investigate further models. Most of them, in fact, cannot approach the boundary configuration shown in Fig. 3.1 and in such a case the application of a first-order accurate BCM like the bounce-back rule drastically degrade the accuracy of the simulation results everywhere [62].

Related work

The current state-of-the-art of BCMs well-suited for curved boundaries were presented by Junk and Yang [170,171]. In particular, in Ref. [170] they devised a velocity BCM which is second-order accurate (in space) for the velocity and first-order accurate for the pressure. The model is local, that is, does not need neighbour information and can handle the topological configuration of Fig. 3.1 but has three caveats. First of all, it is a generalisation of the bounce-back rule which is not very accurate even in the most favourable boundary geometry (the boundaries are half way through the lattice vectors and perpendicular to a Cartesian axis) where it is supposed to yield second-order accurate results for the velocity field (see discussion and results of Chapter 6). Its implicit version is the most attractive one in terms of stability and accuracy but is very computationally demanding. Finally, it is particularly complex to implement. The pressure BCM presented by Yang [171] is also very complex and does not convince us in implementing it within a parallel code because of the programming difficulties inherent in handling communication of neighbour data needed to handle its special boundary-related treatments. In Chapter 6, we have tested it in conjunction with the bounce-back rule in a boundary setup favourable in terms of accuracy and mathematical formulation for implementation purposes; we anticipate that, in this geometrical situation, our BCMs are competitive whilst being much simpler.

Ref. [170] showed that several other BCMs are second-order accurate (for the velocity) in space and all of those which do not require second nearest neighbour information reduce to the bounce-back method in terms of mathematical formulation when the velocity boundary is parallel to a Cartesian axis and is half-way through the lattice vectors; in this case, all their computation, accuracy and stability are the same. Since we found that the bounce-back method is always fairly inaccurate, we were motivated to develop and investigate new ideas. Below, we discuss some new BCMs but we will stress that, as further discussed in Chapter 6, more research should be conducted to fully exploit the intrinsic accuracy of the LBM.

Mathematical description

Here, we continue by describing our novel BCMs; we anticipate that one needs to comprehend Sec. 2.2.6 and Fig. 2.2 of Chapter 2 in order to properly understand the following discussion.

The core notion of the new BCMs stems from the subdivision of the distribution function into the equilibrium and non-equilibrium parts and their approximation at time $t + \Delta t$ with those at time t by means of extrapolation:

$$f_i(\mathbf{x}, t + \Delta t) \equiv f_i^{(eq)}(\mathbf{x}, t + \Delta t) + f_i^{(neq)}(\mathbf{x}, t + \Delta t), \quad (3.4)$$

$$f_i^{(*)}(\mathbf{x}, t + \Delta t) = f_i^{(*)}(\mathbf{x}, t) + O(\Delta t) \quad \text{or} \quad (3.5)$$

$$f_i^{(*)}(\mathbf{x}, t + \Delta t) = 2f_i^{(*)}(\mathbf{x}, t) - f_i^{(*)}(\mathbf{x}, t - \Delta t) + O(\Delta t^2), \quad (3.6)$$

where the symbol “*” can be either *eq* or *neq*. Now, following the Chapman-Enskog procedure (see Chapter 2)

$$f_i = f_i^{(eq)} + \epsilon f_i^{(neq)} \quad (3.7)$$

where the Knudsen number ϵ is proportional to Δx and Δt . As such, the error connected to $f_i^{(neq)}$ in the computation of f_i and the macroscopic flow fields is $O(\Delta t^2)$ if the former is approximated with a first-order extrapolation scheme. Similarly, the error in the macroscopic flow fields introduced by omitting $f_i^{(neq)}$ is $O(\Delta t)$.

$f_i^{(eq)}(\mathbf{x}, t)$ is calculated through $\rho(\mathbf{x}, t)$ and $\mathbf{u}(\mathbf{x}, t)$ with an error proportional to Ma^2 (see Chapter 2). Unfortunately, in general the boundary does not pass through \mathbf{x} but their mutual minimum distance can be any between 0 and Δx . As a consequence, for complex boundaries, a big obstacle to the development of an accurate evaluation of the inward pointing $f_i(\mathbf{x}, t + \Delta t)$ may be the contribution due to the approximation of $f_i^{(eq)}(\mathbf{x}, t + \Delta t)$ since one has to commit errors also dependent on Δx .

HemeLB is currently incorporating the simple local BCM called BCM_{HemeLB} :

$$f_i^+(\mathbf{x}, t) = f_i(\mathbf{x}, t) = f_i^{(eq)}(\tilde{p}, \tilde{\mathbf{u}}) \quad \forall i, \quad (3.8)$$

$$f_i(\mathbf{x}, t + \Delta t) = f_i^{(eq)}(\tilde{p}, \tilde{\mathbf{u}}) \quad \forall i : \mathbf{x} - \mathbf{e}_i \notin \Omega, \quad (3.9)$$

$$f_i(\mathbf{x} + \mathbf{e}_i, t + \Delta t) = f_i^+(\mathbf{x}, t) \quad \forall i : \mathbf{x} + \mathbf{e}_i \in \Omega. \quad (3.10)$$

The first and third equation represent a non standard pseudo-collisional state plus a native streaming stage. In the second equation, the last condition refers to the inward pointing (unknown) lattice directions. For a pressure BC with boundary pressure \bar{p} , $\tilde{\mathbf{u}} \equiv \mathbf{u}(\mathbf{x}, t)$ and we extrapolate in space by assuming $p(\mathbf{x}, t) = \bar{p}$. In this case, the extrapolation error is proportional to the distance between \mathbf{x} and the boundary position $\bar{\mathbf{x}}$ times the pressure gradient or Ma^2 :

$$p(\mathbf{x}, t) - p(\bar{\mathbf{x}}, t) = O(\Delta x \nabla p) \propto \Delta x \text{Ma}^2. \quad (3.11)$$

Analogously, for a velocity BC with boundary velocity $\bar{\mathbf{u}}$ we have $\tilde{p} \equiv p(\mathbf{x}, t)$ and we impose $\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}$ with an error proportional to the lattice spacing and Ma :

$$\mathbf{u}(\mathbf{x}, t) - \mathbf{u}(\bar{\mathbf{x}}, t) = O(\Delta x \nabla \mathbf{u}) \propto \Delta x \text{Ma} \quad (3.12)$$

Note that a lattice site can be close to a velocity boundary and to a pressure one at the same time. As discussed above, because of the absence of the contribution due to the non-equilibrium distribution functions (which yields an error proportional to Δt), the overall BCM_{HemeLB} 's error is generally $O(\Delta t) + O(\Delta x \text{Ma}^2)$ for a pressure BC and $O(\Delta t) + O(\Delta x \text{Ma})$ for a velocity one.

We have implemented a very simple and efficient scheme of the present BCM. In the pre-processing stage, we initialise the elements of f_{dest} like those of f_{source} . Then, for a boundary lattice site \mathbf{x} we replace the elements of $f_{source}(\mathbf{x}, i)$ with $f_i^{(eq)}(\tilde{p}, \tilde{\mathbf{u}})$ for all i and we propagate the populations whose lattice vector does not cross the boundary:

$$f_{source}(\mathbf{x}, i) = f_i^{(eq)}(\tilde{p}, \tilde{\mathbf{u}}), \quad \forall i, \quad (3.13)$$

$$f_{dest}(\mathbf{x} + \mathbf{e}_i, i) = f_{source}(\mathbf{x}, i) \quad \forall i : \mathbf{x} + \mathbf{e}_i \in \Omega, \quad (3.14)$$

This BC scheme slightly differs from its original formulation proposed earlier; here, in fact, the inward pointing distribution functions are not explicitly set at time $t + \Delta t$ but remain equal to the post-collisional values as computed at time $t - \Delta t$. We verified that the two versions yield identical velocity and pressure fields up to the first ≈ 5 digits. Notably, the last equation can be handled without invoking any conditional statement thanks to the algorithmic trick presented in Sec. 3.2 which copies the value of $f_{source}(\mathbf{x}, i)$ to a “ghost” buffer element $\forall i : \mathbf{x} + \mathbf{e}_i \notin \Omega$.

BCM_{HemeLB} is based on a first-order accurate extrapolation scheme in time and space, is very simple and efficient, and can handle any boundary configuration. Unfortunately, it decreases the intrinsic (second order) accuracy of the LBM (see Chapter 2) and introduces an error proportional to Ma –instead of Ma^2 – for velocity-controlled boundaries.

Now, we present a variant of the BCM discussed earlier called BCM^+_{HemeLB} . The post-collisional distribution functions are always calculated as carried out in the LBM. The BCM consists of the first of the following equations (the second one is a standard streaming step):

$$f_i(\mathbf{x}, t + \Delta t) = f_i^{(eq)}(\tilde{p}, \tilde{\mathbf{u}}) + f_i^{(neq)}(\mathbf{x}, t) \quad \forall i : \mathbf{x} - \mathbf{e}_i \notin \Omega, \quad (3.15)$$

$$f_i(\mathbf{x} + \mathbf{e}_i, t + \Delta t) = f_i^+(\mathbf{x}, t) \quad \forall i : \mathbf{x} + \mathbf{e}_i \in \Omega, \quad (3.16)$$

For a pressure boundary, in contrast to the previous BCM where $p(\mathbf{x}, t)$ is assumed to be equal to \bar{p} , here we linearly interpolate $p(\mathbf{x}, t)$ by considering \bar{p} and the pressure at a nearest neighbour site, namely $\hat{\mathbf{x}}$:

$$p(\mathbf{x}, t) = \frac{\bar{p} + qp(\hat{\mathbf{x}})}{1 + q} \quad (3.17)$$

where q is the distance in lattice vector unit (in the range $[0, 1]$) between \mathbf{x} and the intersection between the pressure boundary and the lattice vector $\mathbf{x} - \hat{\mathbf{x}}$. $\hat{\mathbf{x}}$ is chosen so as to maximise the scalar product between the boundary normal and $\mathbf{x} - \hat{\mathbf{x}}$. A velocity boundary is treated in the same way.

It is worth noting that, in contrast to BCM_{HemeLB} , this BCM is not local and involves the information of one or two neighbouring lattice sites. Specifically, a lattice site which is close to the pressure boundary as well as the wall generally requires to load the macroscopic flow field located at two neighbours.

In contrast to BCM_{HemeLB} , the extrapolation in space of the macroscopic flow fields is substituted by an interpolation and is of a second-order accuracy –not of a first-order one; therefore,

$$p(\mathbf{x}, t) - p(\tilde{\mathbf{x}}, t) = O(\Delta x^2 \nabla p) \propto \Delta x^2 \text{Ma}^2, \quad (3.18)$$

$$\mathbf{u}(\mathbf{x}, t) - \mathbf{u}(\tilde{\mathbf{x}}, t) = O(\Delta x^2 \nabla \mathbf{u}) \propto \Delta x^2 \text{Ma}. \quad (3.19)$$

The error in time is $O(\Delta t)$ since the inward pointing lattice directions $f_i(\mathbf{x}, t + \Delta t)$ are approximated with the values of the equilibrium and non-equilibrium distribution functions obtained at time t .

3.2 The HemeLB computational core

In this section, we present the data layouts and algorithmic optimisations pertaining to the computational core of HemeLB; specifically, these data organisations (a) minimise memory footprint, (b) improve data locality, (c) avoid complex access to buffer elements and (d) computational branching. The consequent single-processor programming engine is very optimised (see Sec. 3.4.1).

3.2.1 Preliminaries

A sparse and complex system may occupy a small fraction of its bounding box. For example, a highly complex vasculature is distributed in a space which may be less than 1% of its bounding volume; of course, the LB simulation should only compute tasks related to the volume within such a vasculature to be efficient. Moreover, the LB advection stage entails handling copying the data of one spatial location to another.

A naïve approach to handle a LB simulation of a sparsely distributed system is to resort to a three-dimensional array which corresponds to its bounding volume. Each element of this array is a lattice site and represents a computational unit (voxel) and a position in space. During the simulation, the voxels which do not spatially

correspond to any vasculature volume, called “solid voxels” (or solid lattice sites), can be skipped by employing a buffer –properly set up during the pre-processing stage– which addresses the fluid voxels only. Additionally, it is trivial to propagate data over a Cartesian grid. However, the total memory cost might be unfeasible.

A common strategy to approach LB simulations of a sparse system is to rely on the so-called indirect memory addressing scheme. Essentially, this technique only stores the information concerning the fluid voxels within two one-dimensional arrays (one for f_{source} and one for f_{dest} , see Chapter 2) and the connectivity between their elements and needed during the advection step is maintained by means of another buffer. In the pre-processing stage, the connectivity buffer is built through a lookup table *i.e.* a three-dimensional array which maps a point in space to the corresponding element of the aforementioned one-dimensional arrays. This lookup table can be too memory costly. Additionally, the connections between the one-dimensional buffers can be highly irregular which penalises the efficiency of cache-based processors if f_{source} and f_{dest} are large in size. Last, fluid voxels at the inlets or outlets and close to the wall must be approached with different algorithms which usually forces one to call several conditional statements. Below, we describe how we circumvent these memory and performance related issues.

3.2.2 Handling sparse systems

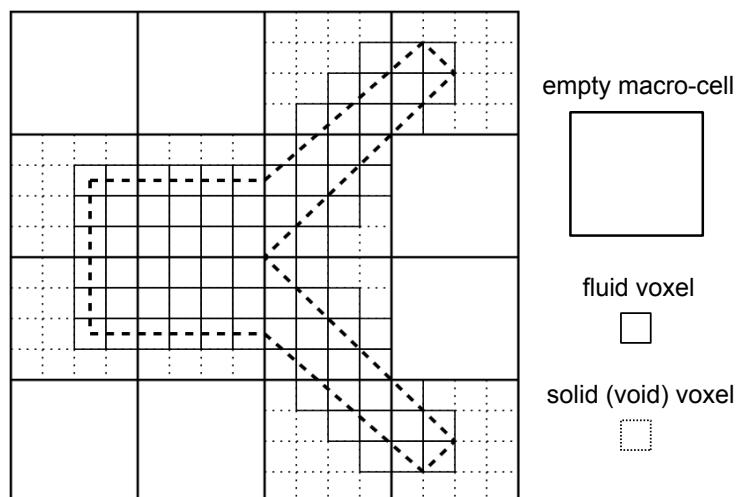


Figure 3.2: Two-dimensional representation of the two-level hierarchical data organization of an idealised bifurcation (thick dashed edges). The bounding box comprises 4×4 macro-cells. Its fluid and solid lattice sites of the Cartesian computational grid are depicted by means of voxels with solid and dashed edges respectively. Only the data associated with each non-empty macro-cell are allocated in memory.

We present a simple data structure which is adapted to sparse systems by avoiding retention of almost all information about the void component whilst achieving good data locality. The grid of the problem domain is split into a two-level representation: if any fluid lattice site is present within a cell of the coarse grid (macro-cell or “block”), this is decomposed into a finer one as illustrated in Fig. 3.2. Multi-level and hierarchical grids have previously been applied to lattice-Boltzmann simulations [124, 128, 172–174]. These approaches have been used in the context of grid refinement techniques to capture with sufficient accuracy small-scale hydrodynamic behaviour within one or more subdomains of the system. No data are exchanged between different grids belonging to the same level of resolution, but are transferred to their parent coarser grids. In our approach, instead, a two-level data structure is used to represent a domain with a single (unique) resolution. No data exchange takes place between the coarser (parent) grid and the finer one: the distribution functions are propagated within the latter only. This two-level structure saves memory with respect to the “full matrix” representation when one or more parent cells are completely occupied by solid lattice sites since corresponding finer grids cannot then be allocated. Furthermore, the indices of the lattice sites of a block are similar; this means that the corresponding data are usually placed together in cache memories. We work with the lattice sites of a block before stepping to another one and, since the data to be exchanged during the LB advection step and between different blocks are less than the others, we achieve some degree of spatial coherence. Specifically, for one single loop over the lattice sites of a block with side b in lattice units, the ratio between those interfacing with neighbouring blocks and the interior ones scales as $1/b$. On the other hand, cache utilisation is likely to become worse by increasing the block size, as occurs in medium and large regular systems.

Multi-level grids with more than two levels can be adopted. However, we found that a two-level grid with $b = 8$ represents a good tradeoff in terms of simplicity, cache use and adaptivity to sparse systems. With this choice, if the coordinate along the X axis of a lattice site is i , those of the corresponding block and its site are $bi = i \gg 3$ and $si = i - (bi \ll 3)$ respectively. Here, \ll and \gg shift the data to the left and right respectively; thus $n \ll 1 = n * 2$, $n \ll 2 = n * 4$ and so on.

We have two alternatives to store f_{source} and f_{dest} . The first strategy, called “direct two-level” is to format them as two-level arrays and to directly accomplish the streaming step as described above; unfortunately, the mapping between different array elements and discussed above is expensive to compute.

The second approach organises f_{source} , f_{dest} and the connectivity buffer needed during the propagation step through the indirect memory addressing scheme discussed earlier. The two-level data grid (three-dimensional lookup table) is only employed to build the connectivity buffer, which saves memory consumption during the pre-

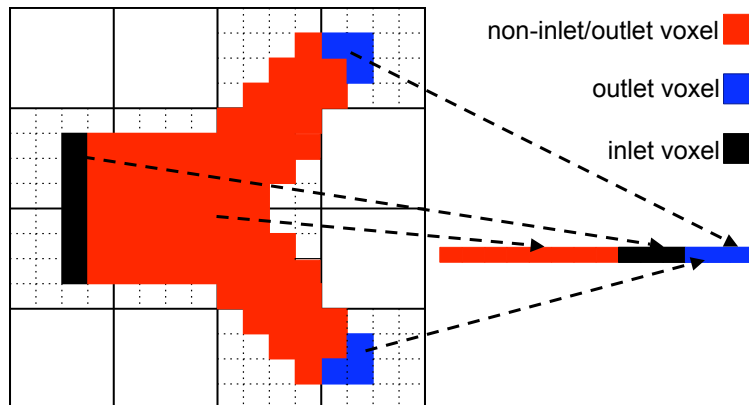


Figure 3.3: Two-dimensional illustration of the two-level data layout (left image) set up and used during the pre-processing stage only, and reordered data format of f_{source} or f_{dest} (right image) employed by HemeLB during the simulation. It is worth noting that data related to different voxel types are kept separate in f_{source} and f_{dest} . See text for further details.

processing stage. This data layout –currently implemented in HemeLB– is shown in Fig. 3.3 and avoids computing the indices of the two-level arrays during the streaming step of the previous data layout implementation (direct two-level).

Additionally, some programming tactics, which we describe below, completely circumvent computational branching and minimise data pattern irregularity.

At first, the streaming towards solid lattice sites appears to be avoided only by including a conditional within the innermost loop indicating whether the neighbouring lattice site is fluid or not. However, the conditional may be eliminated by streaming every particle which would propagate towards a solid site to an unused ghost element of the destination one. If N is the number of fluid lattice sites, the ghost buffer element is covered by the $(N + 1)$ -th one. The source buffer must also be sized with $N + 1$ elements, since at the end of the LB time step they must be swapped (see Chapter 2).

Second, handling different types of fluid lattice sites, *e.g.* interior or wall ones or those residing at the inlets or outlets, appear to entail the use of an (expensive) inner conditional. This may be avoided if, for instance, one regroups the one-dimensional arrays in subgroups: the first N_1 fluid sites will be (interior) non-boundary fluid sites, the next N_2 fluid sites are adjacent to the solid region and so on. This data reorganisation, which is depicted in Fig. 3.3 is carried out during the pre-processing stage where the one-dimensional connectivity buffer is set to properly handle the streaming

stage. Furthermore, the fluid sites of a subgroup are located in close proximity and sequentially addressed by a specialised LB routine and no computational branching takes place.

To sum up, all these data structure and algorithmic optimisations minimise memory footprint, enhance cache and pipeline use, and engender high overall performance.

3.3 Parallel implementation

In this section, we present the parallelisation strategy adopted in HemeLB (Sec. 3.3.1) which is shown to be well-suited to simulate fluid flow in large and complex systems. The new domain decomposition approach is found to be fast and of high quality (Sec. 3.3.1 and 3.4.2). The computational performance achieved with the adoption of the algorithmic optimisations and domain partitioning technique presented in Sec. 3.2 and Sec. 3.3.1 respectively is reported in Sec. 3.4.

3.3.1 Domain decomposition

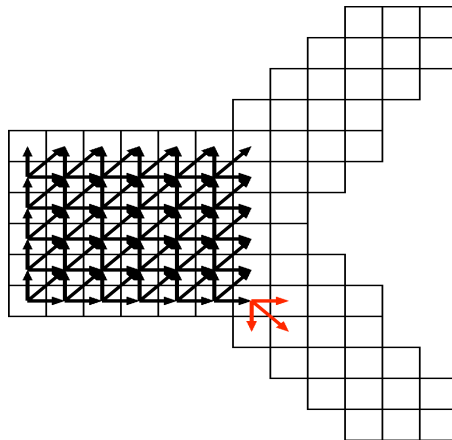


Figure 3.4: Two-dimensional illustration of the domain partitioning algorithm taking place within a voxelised bifurcation. The first processor sub-domain (voxels covered by black arrows) is created by expanding a volume region of fluid voxels through the computational lattice from the left- and bottom-most fluid voxel. When N/P (N and P are the number of fluid voxels and processors respectively) fluid voxels are assigned to the first processor, a new propagation through the lattice, which will generate the sub-domain to be assigned to the second processor, starts from another fluid voxel (red arrows), and so on. See text for further details.

Parallel lattice-Boltzmann codes well-suited for complex systems often employ graph partitioning techniques such as the multilevel k -way partitioning scheme [148,

163] or the ORB approach [142,165,175] to decompose the computational domain into a set of high quality sub-domains so as to achieve good workload and communication balancing during the parallel execution.

Here, a domain decomposition approach based on a graph growing partitioning (GGP) algorithm, a specific branch of the general category of graph partitioning techniques, is presented. GGP partitioning techniques have been presented and used in the past [176–178], but have never been adopted in LB simulations. Basically, the GGP technique grows P partitions from P corresponding seed points by carrying out a breadth-first expansion algorithm [142,142,165,175]. At the beginning of the GGP scheme, all the fluid lattice sites are marked as “unvisited” and one of them is selected and marked as “visited”, and assigned to the processor with identifier (or rank) 0. An iterative search starts from this point and grows a region around it until the desired number of (visited) fluid lattice sites N/P is reached, as explained below and illustrated in Fig. 3.4. The expansion proceeds by means of two “coordinate” buffers that are used to locate the fluid lattice sites at the current and next iteration respectively. For example, initially one coordinate buffer points to the first selected visited fluid lattice site only, while the second buffer will contain the coordinates of the unvisited neighbouring ones. At the end of every iteration the two aforementioned buffers are swapped. The propagation from every fluid lattice site in the first coordinate buffer to the neighbouring unvisited ones is constrained to follow the velocity directions in the LB model used. When the number of fluid lattice sites reaches N/P , $P - 1$ similar searches are performed. Particular care must be taken if a propagation cannot continue because no further unvisited fluid sites surround the visited ones but the sub-domain size is not N/P yet. In this case, another iterative search is started from an unvisited fluid lattice site.

Next, we outline some implementation aspects. In general, each fluid lattice site is checked more than once: the iterative search must start from an unvisited one which is not known *a priori*; the expansion does not proceed in a predefined order and, inevitably, entails repeatedly checking the status of the lattice sites. A dynamic look-up table of the unvisited fluid lattice sites and the velocity directions from which the others originate speeds up the iterative propagation by minimizing repeated accesses at the expense of an increase in memory consumption. However, the algorithm updates $O(10^7)$ points per second on a modern processor core, and the memory consumption is $O(N)$ bytes. Furthermore, perfect load balancing is achieved: the difference in fluid lattice sites between the smallest and largest partition is 0 or 1 if N is multiple of the number of partitions or not, respectively⁴.

⁴In [169] we presented a coarse-level-based GGP approach (instead of the fine-level-based one discussed in this section), which relies on the two-level representation and overcomes the limitations related to the global data requirements and inherent speed but does not guarantee perfect load balancing. The programming choice pursued has been dictated by the fact that the systems under

Two-dimensional representations of the result of the proposed domain decomposition method with 16 partitions applied to a square and a bifurcation with 128^2 and 61451 fluid lattice sites respectively are provided on the right hand side of Fig. 3.3.1, while the Wang *et al.* method [5] produced the images on the left hand side. The grey-scale helps one to recognise the identifier of each partition. The two methods always yield perfect load balancing. However, the shape of the partitions influences the communication balance. With the new partitioning method the minimum and maximum number of interface-dependent fluid lattice sites, *i.e.* the fluid sites adjacent to neighbouring processor sub-domains, are halved and thus inter-processor communications are drastically reduced. Adopting the method discussed by Wang *et al.* [5] the minimum and maximum number of interface-dependent fluid lattice sites are respectively 128/63 and 256/124 for the square and 190/123 and 2370/403 for the bifurcation of those attained with the new GGP approach⁵. From Fig. 3.3.1, Wang *et al.* method [5] appears a generalisation of the slab-based partitioning technique for sparse systems and inherits the latter's communication imbalance even when P is very low with respect to N . Conversely, the success of GGP techniques, including that presented here, relies in the nearly-spherical propagation of the visited fluid sites at each iteration⁶. A good computation and communication balancing, as achieved using the GGP algorithm, plays an important role since a fast computational core, as in HemeLB, makes the effect of large and unbalanced communications even more critical [5].

The parallel LB scheme presented by Schulz *et al.* [148] has been adopted. Briefly, in this parallelisation strategy, after the collision of the interface-dependent fluid lattice sites the propagation of their particles to the neighbouring processors is initiated by means of non-blocking communications. Completion of these communications is enforced only after the computation of interface-independent fluid lattice sites and the data received from neighbouring sub-domains are copied into the destination buffer of the distribution functions, which is swapped with the source one at the end of the LB time step. This parallel scheme can overlap inter-processor non-blocking communication with the computation of the interface-independent fluid lattice sites. This result may be particularly beneficial since the communication may be partially or

investigation are not enormous; in this case it is more important to increase load and communication balancing during the simulation than saving pre-processing memory and speed.

⁵The differences in the values are also due to the presence of system boundaries and hence, the inner partitions are more likely to share a larger number of distribution functions with the neighbouring ones.

⁶In general, the precise shape of the advancing front involved in the GGP search depends on the graph to be partitioned, the presence of obstacles and of visited graph vertices. At each iteration of the GGP search, the visited fluid lattice sites involved at the previous step are arranged in a cube for the D3Q15, D3Q18, D3Q19 and D3Q27 LB models if the propagation does not encounter any obstacle.

completely hidden. Interface-dependent and independent fluid sites are not irregularly referenced through extra arrays but sequentially by reorganising them into two different sub-groups of the same array, adopting the same algorithmic trick presented in Sec. 3.2.

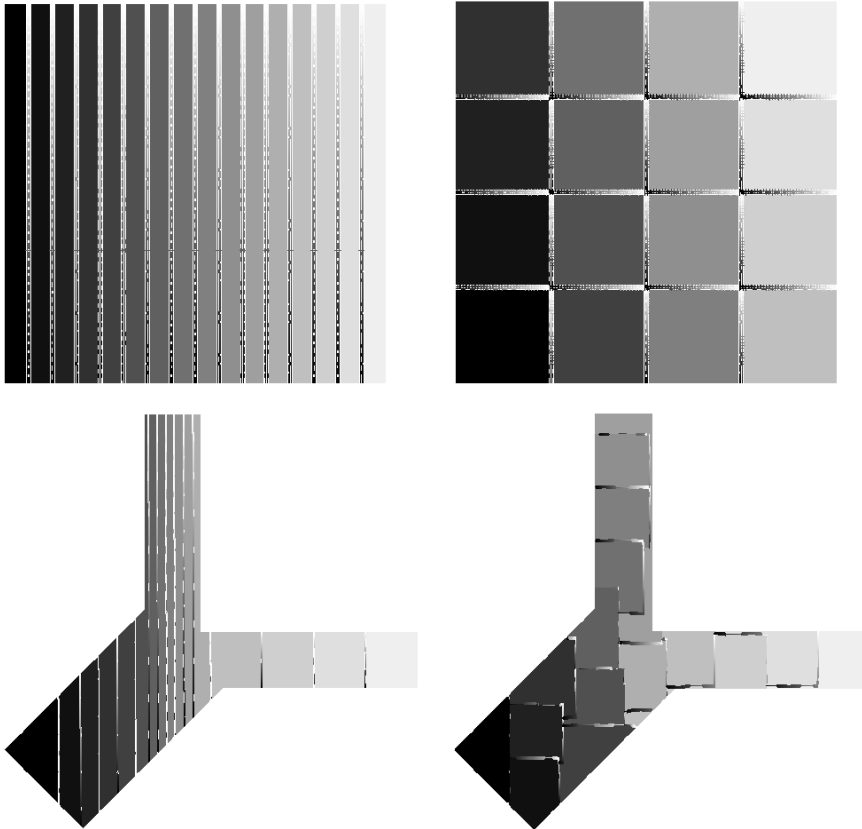


Figure 3.5: Two-dimensional partitions of a square and a bifurcation with 128^2 and 61451 fluid lattice sites respectively, as obtained with the domain decomposition method of Wang *et al.* [5] (left hand side images) and that presented in Sec. 3.3.1 (right hand side). In each image the number of partitions is 16 and their ranks are coloured according to a grey-scale map. The identifiers of the interface-dependent distribution functions between every pair of adjacent partitions are represented by tiny grey segments. The grey intensity is proportional to the identifier. Adjacent distribution functions have the same identifier and cannot be communicated during each time step.

3.3.2 Parallel programming optimisations

We now discuss optimisations incorporated in HemeLB pertaining to parallel programming techniques. The resulting code minimises the amount of intra-machine communication yet hides the complexity inherent in the simulation of sparse systems; moreover, it optimises inter-machine communication in cross-site runs in grid

deployments during each time step.

Minimisation of intra-machine communications

Sparse and complex systems present various difficulties in terms of parallel implementations. Generally, following domain decomposition, the identifiers of the distribution functions to be communicated and the location of the corresponding fluid lattice sites must be determined since they are not known *a priori*, as would be the case for regular domain decompositions. One parallel approach which does not need this step is that of Wang *et al.* [5]. Communicating the identifiers doubles⁷ the size of inter-processing communications at each time step. In HemeLB this and other possible intermediate buffers and memory load/store operations are avoided. Specifically, only the distribution function values must be exchanged between processors and the data to fetch/store from/to the buffers to communicate are solely the ones which are copied from the receiving buffer and stored into f_{dest} ; moreover, this stage involves a single-pass only. The technique to handle these optimisations is outlined below.

The streaming of particles between two different processor sub-domains takes place in three steps: (a) copying of the distribution functions from the source buffer f_{source} to that to send to the neighbouring processor; (b) copying of the latter buffer into the receiving one $f_{to.recv}$ by means of a point-to-point communication; (c) copying of the distribution functions from the receiving buffer $f_{to.recv}$ to the destination one f_{dest} . Steps (a) and (c) represent the difficulty that must be faced if one wants to avoid the use of extra buffers corresponding to multiple indices, because the interface-dependent fluid lattice sites are not arranged regularly and therefore are not known *a priori*. The connectivity buffer employed to load the distribution functions from f_{source} to f_{dest} may be used to accomplish step (a), but another buffer should be exploited to indicate if the current distribution function must be copied into f_{dest} or the buffer to communicate. This can be avoided by extending the length of f_{dest} to accommodate the data to send to the neighbouring processors and properly setting up during the pre-processing stage, the connectivity buffer needed to link the elements between f_{source} and f_{dest} . As a consequence, the loop related to the streaming stage automatically copy the particles between intra-subdomain fluid sites and setup the data to communicate. f_{source} is extended to accommodate the data to receive from the neighbouring processors. During the pre-processing stage, each processor calculates the interface-dependent lattice sites and store them in a buffer, labelled f_{id} . An extra buffer of indices, called $f_{dest.id}$ is set thanks to f_{id} during the pre-processing to copy the data received from the neighbouring processors to f_{dest} ; thus step (c) is simply

⁷Usually, each distribution function is stored in 8 bytes; unfortunately, the same amount of memory is usually required to store its velocity direction and the location of the interface-dependent fluid lattice site.

accomplished as follows:

$$f_{dest}[f_{dest_id}[:]] = f_{to_recv}[:], \quad (3.20)$$

where the symbol “:” means that all the elements of the arrays must be referenced. With these tricks we have also optimised the copying of the distribution functions to exchange between neighbouring processor sub-domains.

With the adoption of the features presented in this section, the resulting parallel core becomes very simple and abstracts the programming complexity caused by sparse representations. As a result, it avoids the need to reference buffer elements in complicated ways and the use of costly conditionals, and minimises data pattern irregularity. In conclusion, the optimisations discussed in this section and Sec. 3.2 permit us to maintain a low overall memory consumption and to drastically reduce the execution time (see Sec. 3.4).

Optimisation of inter-machine communication

The exploitation of multiple machines permits us to gather the computational power needed to quickly study a certain problem or to tackle “grand-challenge” problems including very-large-scale fluid flow simulations. Several problems in the biological and physical sciences require computational resources beyond any single supercomputer’s current capacity [179–182]. The advent of cross-site simulations is already taking place today by means of the various geographically distributed MPI interfaces, such as MPICH-G2 [183] and its new version MPIg [184]. An investigation of the feasibility and scalability of cross-site HemeLB simulations using a geographically distributed domain decomposition is presented in Sec. 3.4.3. The communication between remote machines is characterised by a latency one or more orders of magnitude greater than that affecting intra-machine data exchanges, while the inter-machine bandwidth is considerably lower than the intra-machine one. Thus, the optimisation of inter-machine communications is essential for such an approach to be viable.

The GGP technique produces partitions of high quality (see Sec. 3.3.1). To further minimise communication cost it might be useful to regularise the interfaces between sub-domains which may be carried out by recursively subdividing the system into m parts with m small ($O(10)$); this approach is equivalent to the ORB method (see Sec. 3.0.1) if $m = 2$ and the partitions are obtained proceeding for orthogonal bisections. Thus, if the number of machines used is m , the strategy adopted subdivides the system into m macro-partitions. The size of each one is chosen to be proportional to the number of processors of the corresponding machine. A topology discovery mechanism based on the MPI function `MPI_Attr_get` were implemented in HemeLB in order to detect the number of machines used and the number of processors for each machine.

The topology discovery implementation, the two-level GGP decomposition and the adoption of the Schulz *et al.* parallelisation scheme [148] permit us (a) to optimise inter-machine communications so as to hide high latency and low memory bandwidth, and (b) to tune cross-site simulations to specific heterogeneous resource distributions. Unfortunately, with regard to MPIg and MPICH-G2, only MPIg significantly hides inter-machine communications via non-blocking message transmissions that occur between the supercomputing resources. The current parallel implementation of HemeLB does not optimise the domain decomposition to take into account the different performance of the plurality of machines that may be present in a grid. However, this heterogeneity may be taken into account simply by adopting the algorithmic scheme used to consider the different numbers of processors used in each machine, as discussed above.

Fast and cheap convergence criterion

Here, we present the approaches adopted in HemeLB to check the convergence status in stationary and periodically time-varying non-turbulent simulations.

The fluid reaches a stationary flow behaviour if the inlet and outlet pressures are not varying in time and the fluid flow is not turbulent. In this case, it is sufficient to check if the velocity flow field between two successive time steps is invariant within a prescribed tolerance. The velocity flow field at the current time step is compared with that of the previous one, which can be stored in a compact one-dimensional buffer; every element of that array is compared with the current velocity and substituted with its value.

If the simulation is controlled by periodically-varying boundary conditions and the fluid flow does not become turbulent the resulting fluid flow settles down to a certain time-dependent flow field after a certain number of time periods (not known *a priori*). In this case, unfortunately, the application of a convergence criterion is more delicate: confronting the flow fields of two successive time periods, similarly to the previous method, entails storing the velocity flow field associated with an entire period. Therefore, a system comprising 1M fluid lattice sites only and 100K time steps per period would require more than 2000 GB of memory, which is unreasonable. A strategy to drastically reduce the memory footprint is to check convergence through randomly selected spatio-temporal points.

In HemeLB we have implemented another idea: the convergence status is checked by comparing the flow field of the current simulation at the current time step with that obtained by a parallel simulation delayed by one time period with respect to the first one. As a consequence, the memory consumption associated with some buffers is doubled.

3.4 Performance scenario

In this section, the single-processor-core performance of the serial cache-aware LB programming engine adopted in HemeLB is presented and compared to the one directly exploiting the two-level representation. Then, the parallel performance and scalability of HemeLB is discussed. Other benchmark results are given in Chapter 4 together with those achieved with our parallel visualisation method. Finally, cross-site performance results are provided.

3.4.1 Single processor performance

Here, we present the single-processor performance of HemeLB which is compared to that achieved by employing the two-level data representation explicitly. Essentially, the purpose of this comparison is to verify the benefits associated with the optimisations described in Sec. 3.2 (optimised buffer access, pattern regularity and computational branching).

We carried out the benchmarks by applying the D3Q15 model and boundary condition method presented in Sec. 3.1 to a set of bifurcations of different sizes. The simulation cells are composed of 16^3 , 32^3 , 64^3 , 128^3 , 256^3 and 512^3 lattice sites with 225, 1601, 12449, 98211, 780359 and 6253409 fluid lattice sites respectively. The pressure profile of the largest system is shown in Fig. 3.6.

The benchmarks were obtained employing a single core of a Intel Core 2 Quad 2.5 GHz with 3 MB cache/core running the operating system Linux Fedora 8 64 bits, and the compiler Intel 9.1 with flags `-O3`, `-ipo`, `-xT`. The performance in MSUPS (millions of fluid site updates per second) is shown in Fig. 3.7 for the pure HemeLB implementation and the HemeLB's version which directly employs the two-level data representation. Specifically, the second approach does not exploit the algorithmic tricks which minimise pattern irregularity, complex buffer accesses and computational branching, as outlined in Sec. 3.2.

The performance attained by HemeLB is superior to that achieved by the highly tuned code developed by Donath *et al.* [144] and benchmarked on architectures similar to ours: 8-20 versus 5-7 MSUPS. The pure implementation is more than two times faster than its version which directly employs the two-level data representation (see Sec. 3.2 for details), especially during the simulation of small systems; this demonstrates the enormous improvement due to the algorithmic tricks, presented in Sec. 3.2, which optimise complex buffer access, data pattern regularity and avoid computational branching⁸. The ratio between wall fluid sites and the other ones is greater for smaller systems. Since the wall fluid sites are handled with less operations

⁸In [169] it is also shown that data reordering techniques, as the two-level representation implemented in HemeLB, can produce higher performance than typical cache-aware optimisations such as loop blocking due to better cache utilisation.

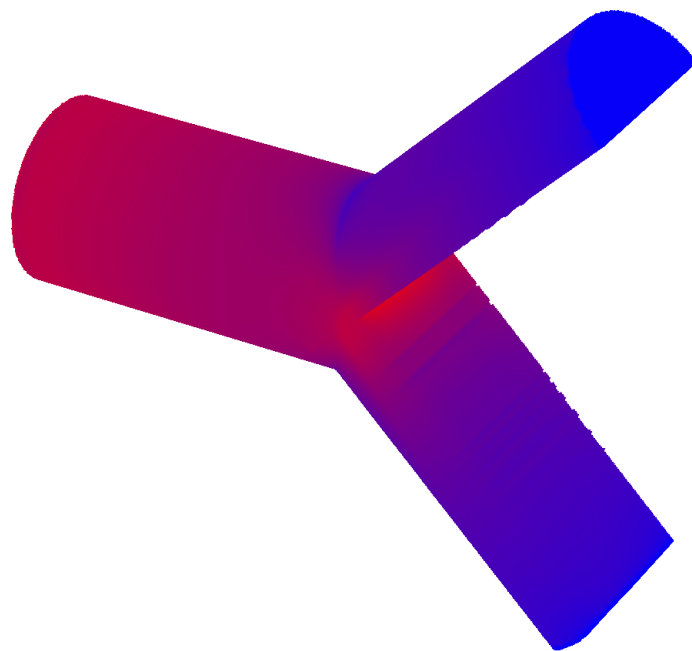


Figure 3.6: External pressure profile of the stationary flow field pertaining to a bifurcation of 6253409 fluid lattice sites. The wall and pressure boundary conditions applied to the no-slip walls and inlet/outlet lattice sites respectively are discussed in Sec. 3.1. Red and blue colours represent maximum and minimum pressure respectively. The colour of any fluid lattice site is linearly interpolated between those values according to its pressure.

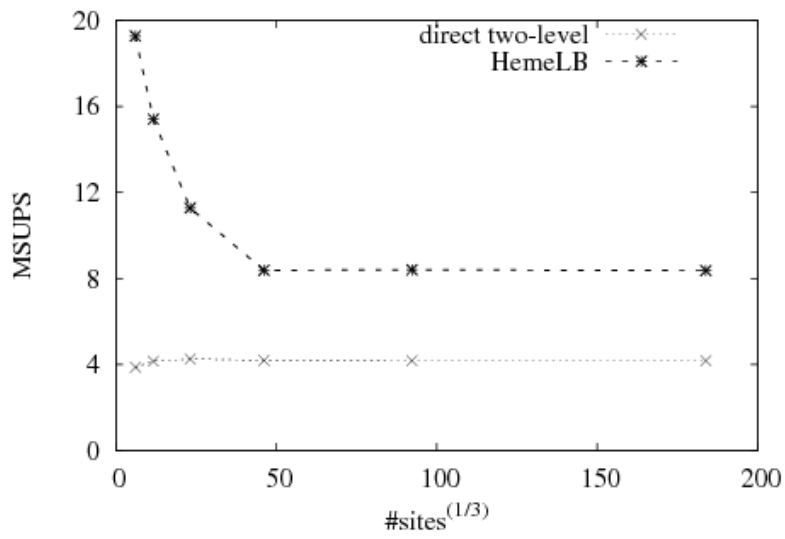


Figure 3.7: Performance measured in millions of fluid site updates per second (MSUPS) as a function of the cube root of the fluid lattice sites using HemeLB and a HemeLB’s version (‘direct two-level’, see Sec. 3.2 for details) which explicitly adopts the two-level data representation of HemeLB and therefore is cache-aware but does not exploit the other optimisations presented in Sec. 3.2. The timing results were conducted on a single core of a Intel Core 2 Quad 2.5 GHz with 3 MB cache/core running the operating system Linux Fedora 8 64 bits; the compiler used was Intel 9.1 with flags `-O3, -ipo, -xT`.

than the others (see Sec. 3.1), the aforementioned ratio as well as cache use affect the performance behaviour as a function of system size.

3.4.2 Single-machine parallel performance

In this section, we report some benchmarks related to the parallel HemeLB code presented in Sec. 3.3. Single-site parallel performance has been tested on HPCx, the IBM Power5 1.5 GHz housed at Daresbury Laboratory (UK) [186] and on Abe, the TeraGrid cluster of Intel quad-core 2.33 GHz processors located at the University of Illinois' National Center for Supercomputing Applications (NCSA) [187]. The IBM machine is composed of a number of nodes communicating via the IBM High Performance Federation Switch. Every node is comprised of 8 chips, each of 2 cores that share L2 and L3 caches (see [186] for further details). The compiler flags used are `-qstrict`, `-qstrict_induction` and `-O4`. Each Abe's node comprises 2 Intel quad-core processors, which are interconnected by a Infiniband interface (see [187] for further details). The compiler flags used are `-O3`, `-xT` and `-ipo`.

Benchmarks were conducted on a patient-specific system discretised with 7.7M fluid lattice sites obtained from a medical MRA dataset (Fig. 3.8 shows its pressure profile). The LB model and the boundary condition method presented in Sec. 3.1 were used. The minimum number of processor cores used was 16.

In Fig. 3.9 we report HemeLB's single-site performance in MSUPS as a function of the number of processor cores. All benchmarks were obtained without checking for instabilities or applying convergence criteria (their performance cost is insignificant if they were rarely performed).

We note that HemeLB yields outstanding intrinsic performance and a superlinear speedup behaviour is maintained up to at least 512 and 2048 processor cores using platforms HPCx and Abe respectively. This depends on the fact that for a large processor count each processor sub-domain is small thereby likely residing in caches, which enhances performance. We investigated other systems and domain partitioning methods [169]; there, we also demonstrated that the GGP scheme applied to the coarsest level of the two-level data representation⁹ guarantees far better performance than that achieved using Wang *et al.* method [5], and similar speed to that carried out by adopting an ideal regular domain decomposition when simulating axis-aligned fluid flows. Moreover, it was also shown that the intrinsic performance, measured in MSUPS, does not significantly depend on the system shape.

The time required to accomplish the domain decomposition calculated by the processor core with rank 0 and that required to manage the rest of the pre-processing

⁹The situation should be even more favourable by employing the partitioning scheme at the finest level since the load balancing becomes perfect and the interfaces between adjacent subdomains are more regular and of a higher quality (see Sec. 3.3.1 for further details).

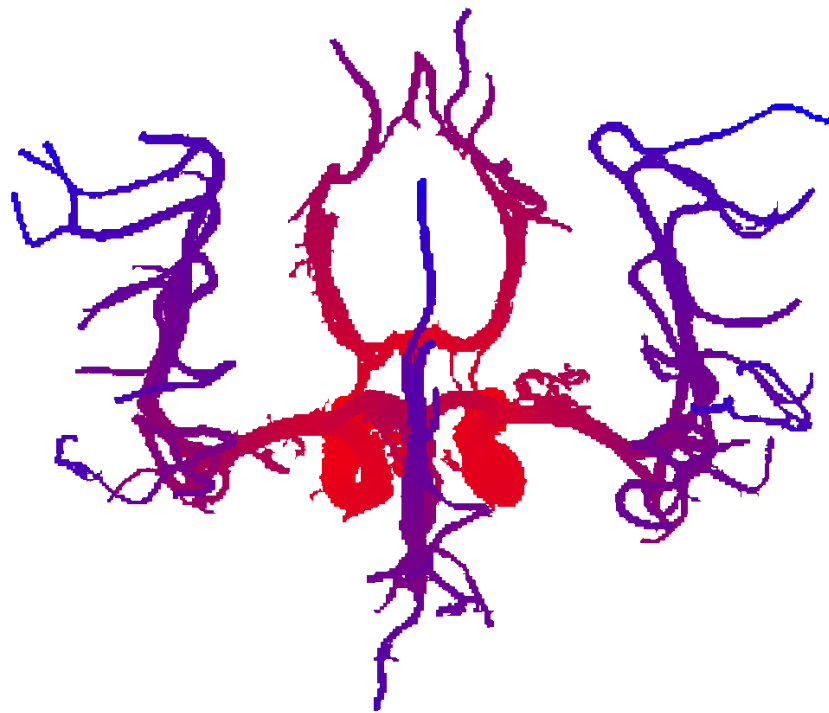


Figure 3.8: External pressure profile of the stationary flow field pertaining to the benchmarked patient-specific system which comprises 7.7M fluid lattice sites. Red and blue colours represent maximum and minimum pressure respectively. The colour of any fluid lattice site is linearly interpolated between those values according to its pressure.

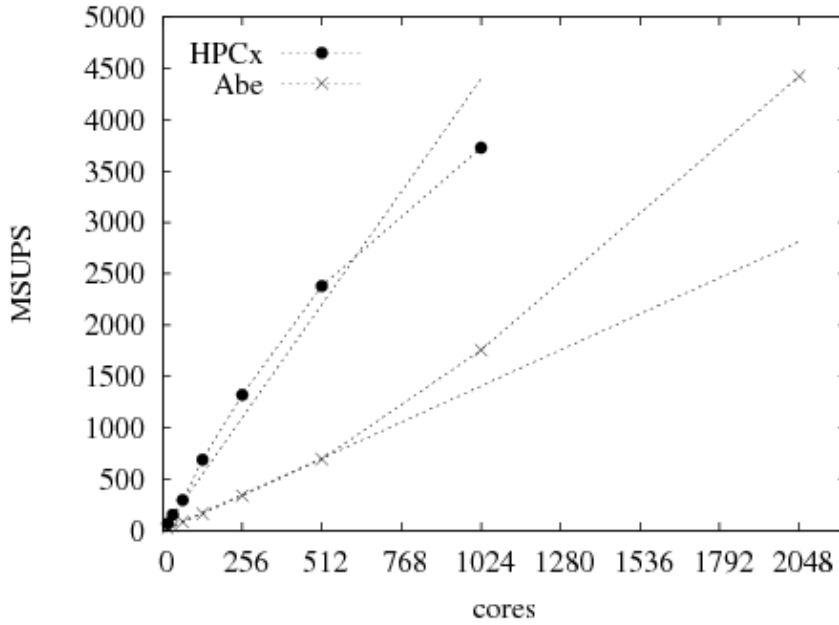


Figure 3.9: HemeLB’s single-site parallel performance measured in millions of lattice site updates per second (MSUPS) as a function of the number of processor cores used for the patient-specific system of 7.7M fluid lattice sites on platforms HPCx and Abe (see Sec. 3.4.2 for further details). Ideal performance is depicted with dashed lines.

stage without considering input reading are only 1.55 and 0.846 seconds on 1024 cores; on 64 cores, instead, those performance numbers are 1.70 and 1.353 respectively.

3.4.3 Cross-site parallel performance

Cross-site benchmarks were performed on the IBM Power5 AIX clusters housed at Louisiana Tech, Tulane University and ULL, called Bluedawg, Ducky and Zeke respectively comprising part of LONI (Louisiana Optical Network Infrastructure). Each node of each cluster comprises 8 IBM Power5 1.9 GHz processor cores. The network interface is a IBM High Performance (Federation) Switch [188]. The compiler flags used are `-O2 -qstrict -qstrict_induction`.

We tested HemeLB on a patient-specific system with 4.69M fluid lattice sites. In Fig. 3.10, the performance in MSUPS as a function of the number of LONI clusters used is provided; 32 processor cores were employed for each cluster and the grid-enabled MPI interface adopted for inter-machine communications is MPIg. Note that high scalability and parallel efficiency are achieved even with the employment of all three clusters; this is an impressive result considering that the simulation is running across geographically distributed platforms.

It is difficult to compare HemeLB’s performance with that of other parallel lattice-Boltzmann codes, developed for complex systems. Each code has been tested using different lattice-Boltzmann models, architectures and flow field systems. However, to

our knowledge, the parallel implementations that yield the best performance results in terms of scalability and overall execution speed are those presented by Wang *et al.* [5] and Axner *et al.* [164]. In [169] it was directly demonstrated that the domain decomposition strategy adopted in [5] is drastically inferior to that incorporated in HemeLB. Moreover, HemeLB exploits several novel optimisations that significantly speed up computation. The fluid solver presented in [164] give a similar performance to that achieved by HemeLB.

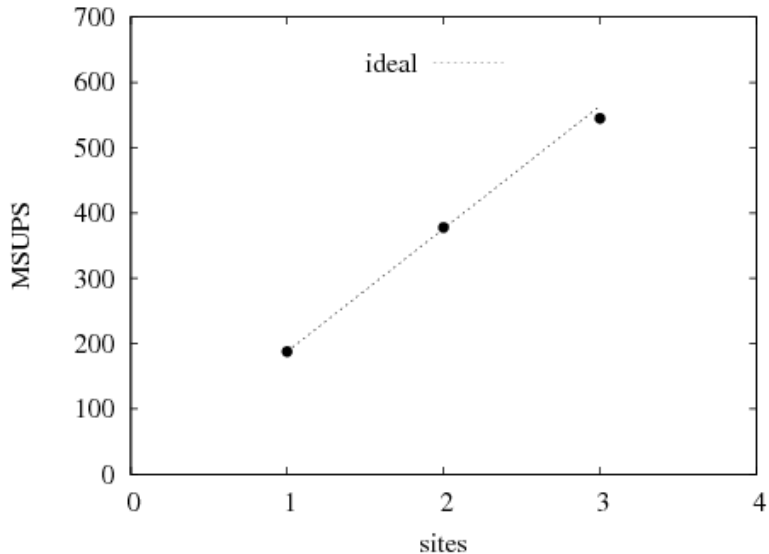


Figure 3.10: HemeLB cross-site performance in millions of fluid lattice site updates per second (MSUPS) as a function of the number of LONI clusters, each contributing 32 processor cores (see Sec. 3.4.3 for further details). Ideal performance is depicted with a dashed line. The fluid flow comprises 4.69 fluid lattice sites. MPIg was employed to handle inter-machine communications. The executable was obtained compiling with flags `-O2, -qstrict -qstrict_induction`.

3.5 Conclusions

We have described a very efficient parallel LB code which is well-suited to study sparse fluid systems. The fast and cheap computational core relies on a two-level data representation and numerous algorithmic optimisations. HemeLB minimises memory consumption when tackling complex sparse systems and yields a faster execution than conventional implementations due to good cache use. Several optimisations were described; they reduce redundant operations, increase pattern regularity, simplify the computational core, eliminate computational branching, further reduce memory consumption and optimise intra-machine communications.

The novel topology-aware domain decomposition is very fast and guarantees high quality partitions. We have demonstrated its capability to effectively decompose a

large system, and its excellent workload and communication balance. Consequently, HemeLB's parallel scalability and performance are very good. The ability to overlap non-blocking communications with computation whenever possible through the use of an optimised MPI implementation plays a central role if maximum computational performance is to be realised. Moreover, a simple modification of the domain decomposition technique permits us to optimise inter-machine communications in grid-based cross-site runs. As a consequence, HemeLB can effectively tackle large and complex systems via distributed heterogeneous computational resources.

One of the main objectives is to use HemeLB for the study of a variety of human cerebral blood flow scenarios, ranging from normal to neuropathological conditions, including aneurysms and arterio-venous malformations, as well as the entire intracranial vasculature; some patient-specific simulation results are provided in Chapter 6. The excellent parallel performance attained using heterogeneous resources is particularly attractive for addressing patient-specific cerebral blood flows in clinically relevant wallclock times (see Chapter 5 and 6). The use of sufficient aggregated computational power permits us to tackle large systems in real time or nearly-interactive rates.

Chapter 4

High performance volume rendering and steering of time-varying fluid flow

To live a creative life, we must lose our fear of being wrong

Joseph Chilton Pearce

Recent algorithm and hardware developments have significantly improved our capability to interactively visualise time-varying flow fields. However, when visualising very large dynamically varying datasets interactively there are still limitations in the scalability and efficiency of these methods. Here we present a rendering pipeline which employs an efficient *in situ* ray tracing technique to visualise flow fields as they are simulated. The ray casting approach is particularly well suited for the visualisation of large and sparse time-varying datasets, where it is capable of rendering fluid flow fields at high image resolutions and at interactive frame rates on a single multi-core processor using OpenMP. The parallel implementation of our *in situ* visualisation method relies on MPI and no specialised hardware support, and employs the same underlying spatial decomposition as the fluid simulator. The visualisation pipeline allows the user to operate on a commodity computer and explore the simulation output interactively. Our simulation environment incorporates numerous features that can be utilised in a wide variety of research contexts.

4.1 Introduction

Volume rendering and isosurface visualisation have played a fundamental role in the analysis of simulation results in the areas of turbulence, multicomponent fluid flow and hemodynamics, amongst others. These visualisation methods provide us with an

accurate and immediate description of complex three-dimensional volumetric data; visualisation can be used to procure an overview of the large-scale dynamical features as well as effective analysis of, for example, physical structures, small-scale phenomena and turbulence patterns. The growth of large scale computational resources and the resolution of data from medical scanning equipment have dramatically increased over the last decade. The rapid and interactive visualisation of such volume data serves as a fundamental tool by means of which scientists can explore and discover natural processes, without sacrificing the fidelity of the source data. The role of efficient visualisation engines is even more important when dealing with time-varying volume data; the total amount of data can easily exceed high-end storage capabilities and its effective visualisation can challenge cutting-edge computational resources.

Here, we focus on time-varying blood flow simulation on high-end computational resources using HemeLB (Chapter 3) to perform fluid simulations within the cerebral vasculature. HemeLB is a lattice-Boltzmann (LB) fluid solver which is designed to simulate steady-state and pulsatile flow in sparse structures such as the human vascular tree. Built into HemeLB is a ray casting algorithm which is the focus of this chapter, whereby the details of the flow can be visualised as the fluid simulation is running. Rather than generating the time-varying data, writing it to disk, then post-processing it for visualisation purposes, we instead render the flow field directly within the fluid simulation session, directly outputting rendered frames to a network connection (for example), avoiding lengthy I/O as well as pre-processing and post-processing times. The isosurface and the volume rendering of time-varying flow field simulation results can be interactively visualised on a remote computer. The interactivity is enabled through efficient steering capabilities such that we can modify simulation and visualisation parameters at run-time, limited only by the network latency between the remote computer and the simulation machine. The resulting interactive simulation environment comprises two applications only: HemeLB with the integrated rendering and network plus steering capabilities, and the application which displays the flow field at run-time or *a posteriori* during a post-processing phase. HemeLB can also write the hemodynamic state to file such that the resulting fields can be rendered by other in-house post-processing tools or commercial visualisation software applications. The overall approach is very useful for the scientist to effectively explore simulation output and to gain insight into the large parameter space associated with such problems.

In this chapter, we begin with an overview of volume rendering techniques in Sec. 4.2, then go on to present the various components of our visualisation pipeline and our performance results in Sec. 4.3 and Sec. 4.4 respectively. We present our conclusions in Sec. 4.5.

The steering approach was implemented by Dr. Steven Manos who also bench-

marked the steering-based interactive runs. Mazzeo contributed to the improvement of the steering algorithm and was responsible for all the other implementations and performance tests.

4.2 Related work

Recently, interactive rendering of large-scale volumetric data has become possible through the efficient use of modern processors, multi-core architectures and algorithm optimisations. Since the increased speed of processing units has surpassed the improvement in memory bandwidth [190], rendering algorithms have focussed on minimising inefficient memory access [191–193]. Data must be exchanged between central processing units (CPU), on-chip memory caches, main memory, hard disk and graphics processing units (GPUs), as well as between different platforms, for example, in the scenario where the rendering is performed on a remote machine which does not have graphics capabilities. In this section, we provide an overview of the literature surrounding volume rendering approaches encompassing the aforementioned aspects with an emphasis on large time-varying datasets. Moreover, we do not discriminate between isosurface and volume visualisations since the former can be seen as a special case of the latter, that is, the visualisation of a certain three-dimensional region.

4.2.1 Visualisation of static volumetric datasets

Several rendering methods have focussed on the visualisation of static volumes. Common to all these methods is pre-processing, a strategy used to accelerate volume rendering [192, 194–196]. Pre-processing involves setting up simple buffers and/or encoding the original data in tailored orderings and data structures such that the processing of unimportant parts of the dataset, which are transparent or occluded by opaque volume regions, is minimised or ignored. Examples of pre-computation include pre-shading and pre-gradient estimation and storage [191, 192]. Efficient transparent region skipping is usually achieved by means of min/max multilevel spatial hierarchies, by using recursive grids [197], kd-trees [196] or octrees¹ [192, 193]. These have also been coupled with more exotic data structures [193]. Hidden volume removal can be effectively carried out by using hierarchical occlusion maps (HOMs) [192] or may be intrinsically handled by the method employed, *e.g.* ray tracing. In some cases, pre-computation may be very time-consuming and require excessive storage which impairs the upper limit of volume size that can be investigated. Extra buffers

¹Kd-trees and octrees are recursive spatial hierarchies. Specifically, in a kd-tree each non-leaf cell is comprised of two disjointed cells whose total spatial extent covers the space of the former. The octree is identical to a kd-tree except for the fact that each non-leaf cell is comprised of four non-overlapping cells.

to accelerate volume rendering may require three to four times the memory space needed by the original dataset [191, 192], which is likely to be unacceptable. Grimm *et al.* [193] circumvented this problem by designing a good compromise between pre-computation and memory consumption by making use of on-the-fly gradient caching and memory-cheap data structures. The min/max recursive grid presented by Parker *et al.* [197] has an insignificant memory overhead; implicit kd-trees drastically reduce storage consumption with respect to explicit ones [196]. On-the-fly computation may also reduce overall rendering time [193, 198] since it mitigates the impact of costly look-up table access.

Volume rendering on GPUs by means of two or three-dimensional textures offers a high-quality interactive approach [199, 200]. Unfortunately, it is not suitable for large datasets due to the limited graphics card memory and memory bandwidth between GPU and CPU which prevents fast out-of-GPU-core approaches. Furthermore, it is difficult to skip transparent regions or take advantage of hidden volume removal effectively [201]. Where the limited memory on current GPUs is not an issue, one could perform the simulation and visualisation on GPU resources only.

Ray tracing is increasingly popular because advanced graphics effects can be built into this technique. Its intrinsic complexity is lower than that of other methods for a large dataset. Specifically for volumetric data, ray tracing has a complexity $O(PN)$ where P is the number of pixels of the screen, and N^3 is the number of voxels. If the technique can take advantage of early ray termination (ERT) [202] so that each casted ray can be stopped after a few volume samples, the sampling complexity becomes $O(P \log(N))$, while the one pertaining to other volume rendering methods is usually $O(N^3)$. The use of min/max hierarchical spatial trees, as discussed above, can drastically accelerate empty space ray traversal. Moreover, ray tracing inherently employs *frustum culling* where volume regions outside the view frustum cannot be seen and are not processed. The effectiveness of view-dependent methods has been widely used in the visualisation of large systems [203]. Techniques based on *multi-resolution level-of-detail* (LOD) schemes [204–206] have been demonstrated to be very beneficial in speeding up rendering whilst maintaining visual accuracy.

Although the ERT algorithm was published many years ago [202], it is still one of the most commonly used algorithms for volume rendering acceleration. Unfortunately, parallel rendering approaches based on a static spatial decomposition of the volumetric dataset (sort-last approaches) are not well-suited for the adoption of ERT since they are not mutually affine, as explained by Matsui *et al.* [207]. These authors presented an algorithm to take advantage of ERT, but their approach entails subdividing processors between rendering and compositing tasks, employing a complex communication pattern where frequent communications cannot always be overlapped with computation.

The state-of-the-art volume rendering of static datasets which does not utilise LOD acceleration is represented by (i) the *shear-image* order ray casting method (augmented with specialised hardware support) [208], which is an improvement over the shear-warp technique [209], and (ii) ray tracing based on kd-trees [196, 206, 210]. The power of shear-image ray casting resides in its high computational and memory coherency and small memory overhead, which are fundamental aspects in the effective utilisation of modern architectures. The efficiency of ray tracing with kd-trees relies on their ability to easily skip empty or transparent spaces. Furthermore, kd-tree traversal algorithms, which take advantage of advanced hardware functionalities (*e.g.* single instruction multiple data (SIMD) instructions), speed up execution when the area of the projection of the volume voxel is comparable in size to the pixel [196].

4.2.2 Visualisation of time-varying volumetric datasets

Algorithms developed for the efficient visualisation of static volumes may be unsuitable for the interactive visualisation of time-varying volumetric datasets since they often rely on optimisations that strongly depend on the dataset itself that now varies over time.

In visualising a time-varying volume, each temporal frame can be considered individually by using one of the aforementioned techniques, or by using approaches that evaluate the temporal series as a whole and capitalise upon building suitable spatial and temporal (4D) data trees. These techniques exploit temporal coherence between data associated with the different time steps. This permits a trade-off between rendering speed and quality [211] and can provide very accurate interactive visualisation of time-varying vector fields by using pathlets mixed with volume rendering, as developed by Yu *et al.* [212]. In these cases, pre-processing carried out at the beginning of the temporal visualisation minimises inter-time-step delay. For large time-varying datasets with several time steps, the overhead of memory access can be an obstacle depending on the platform used. Pre-processing time is typically large and 4D data trees cannot be easily updated if extra time frames are added.

Min/max lossless-compression octrees can be attractive for interactive time-varying volume rendering since they often require a fraction of the original memory footprint [213]. Unfortunately, they require a large pre-processing time; moreover, the storage of long temporal sequences of a large system is prohibitive.

Another interesting category of techniques which have been used for time-varying isosurface rendering does not utilise temporal coherence, instead minimising the impact of I/O by reading from disk the portions of data necessary to construct the current isosurface [214, 215]. To accelerate I/O during rendering, the entire dataset must be properly formatted but, unfortunately, this requires a long pre-processing phase.

Compression and decompression techniques permit one to increase the size of the rendered system and reduce communication between different hardware components, but may be unsuitable for a system that varies rapidly in time since the decompression cost can be too high [216].

Strengert *et al.* [216] report on remarkable work concerning parallel time-varying volume rendering leveraging hierarchical texture data representations compressed with wavelet techniques and rendered with GPUs. An adaptive decompression scheme is adopted on each node before sending data to the local graphics board's texture memory used during rendering; then, the node-dependent sub-images are blended by using the compositing technique of Stompel *et al.* [217]. In general, compression techniques have proved to be very useful in minimising storage requirements and the impact of data transmission between different hardware components. This could permit a dataset to fit in CPU or GPU memory; thus, costly data transfer is not necessary and the rendering is substantially speeded up. If this is not the case, compressed data is transferred into CPU or GPU memory more quickly than its uncompressed counterpart; as a consequence, the employment of compressed data is beneficial for out-of-core approaches whose efficiency strongly depends on the size of the data to transfer between hard disk, CPU and GPU memory [218]. Unfortunately, compression requires one to maintain both the raw and compressed versions of the dataset and it may take seconds to perform encoding in the pre-processing phase and decoding during rendering. However, the cost of the latter is not a problem if performed in GPU hardware [218]. Furthermore, out-of-core rendering solutions that take advantage of the use of multiple processors and parallel I/O further enhance interactivity [219].

Run-time visualisation techniques

An attractive approach to visualise time-varying data produced by a simulation is to render the data whilst the simulation is running – *i.e.* run-time visualisation. This overcomes the limitation in storing the volume data at several time steps (as in a traditional post-processing technique), thus offering the possibility to visualise an unlimited number of time frames and to provide immediate feedback on the ongoing simulation². This is particularly important in the context of computational steering which permits one to adjust simulation parameters on the fly, thus visualising changes immediately, augmenting exploration and scientific discovery [220, 221].

One method to implement run-time visualisation is to frequently transfer simulation data to another platform as described by Yu *et al.* [222], Bellemann and Shulakov [223] and Insley *et al.* [224]. Unfortunately, this methodology requires a high-speed network, parallel I/O and powerful computational resources to be effec-

²Nonetheless, this strategy does not preclude writing to file the flow field at user-selected time steps for conventional post-processing purposes.

tive; furthermore, highly interactive rendering of large datasets is not guaranteed.

An emerging run-time visualisation technique which relies on performing the simulation and rendering within the same platform is called *in situ* rendering. There are two main methods used to handle *in situ* rendering. One is to perform rendering on one processor or a subset of processors which differ from the one employed for the simulation. This type of *in situ* rendering has been described by Rowlan *et al.* [225] and Takei *et al.* [226]. The other strategy to handle *in situ* rendering is a one-to-one kernel of simulation and rendering processes. Specifically, the latter relies on a sort-last method in which each processor renders the subdomain for which it is responsible while the simulation is running. The image is obtained through a compositing stage which merges the processor-dependent sub-images. This *in situ* approach was first developed by Ma [227] and subsequently employed by Lomdahl for ultra-scale molecular dynamical simulations [228] and by Tu *et al.* for terascale earthquake simulations [229]. It is the method we use in the present thesis.

With respect to the previous class of run-time visualisation, this offers the advantage of reducing inter-machine communication such that only the current image is communicated, which may also be compressed, to a host computer equipped with graphics capabilities; alternatively, or additionally, every image may be written to file.

4.3 The visualisation pipeline

The objective here is to interactively simulate and visualise large time-varying volumetric datasets generally discretised by a complex computational ensemble of cubic voxels. The flow field is simulated by HemeLB (see Chapter 3). HemeLB can simulate a time-dependent flow field in a regular or complex geometry, at hundreds of iterations per second by employing a sufficient number of processors. While it is not necessary to monitor a flow field at hundreds of time steps per second, it is undoubtedly useful to visualise the output interactively, *i.e.* 5–20 frames per second (FPS), to enhance exploration and scientific outcome. Real-time volume rendering of large time-varying fields is a challenging problem, as we discussed in Sec. 4.2; in fact, in that section we reported the state-of-the-art of time-varying volume rendering but the best results are achieved at a rate of 5–10 frames per second [216]. Additionally, the storage of large flow fields over hundreds or thousands of time steps is not feasible; the most suitable approach to render several time frames is that offered by *run-time* strategies.

We have adopted the most extreme form of *in situ* visualisation: the rendering is approached by a ray casting technique which has been incorporated in HemeLB such that each processor ray traces its own subdomain, corresponding to the domain decomposition used in HemeLB. We have implemented the ray casting method based on

explicit min/max kd-trees described by Wald *et al.* and Friedrich *et al.* [196,206], one of the fastest techniques to date for time-independent volumes (Sec. 4.2.1). Our simple ray caster relies on a novel data layout particularly suitable for sparse geometries and out-performs the methods presented in [196,206] (see Sec. 4.4). Furthermore, in contrast to these methods, ours does not need pre-processing, thus enabling efficient visualisation of time-varying flow fields. Image compositing is handled by an optimised binary communication pattern. We have also implemented steering capabilities to interactively visualise the flow field in different ways and change its parameters on the fly. The resulting steering and visualisation pipeline permit the fluid solver to run without interruption and at near-maximum speed whilst visual feedback is provided at maximum frequency.

The Message Passing Interface (MPI) [230] is employed to handle inter-processor communication. All the software components, *i.e.* simulation, rendering, image compositing and steering, have been tightly integrated together to maximise performance.

4.3.1 Volume rendering core

We have implemented a simple ray casting algorithm with some optimisations and novel features particularly well-suited for sparse, time-varying flow fields. First, we consider *frustum culling*, *hidden region removal* and *transparent region skipping*, in conjunction with effective volume rendering, and we justify the use of a ray tracing method. Second, we provide the technical details pertaining to the basic and novel features, and optimisations of our ray casting approach.

Frustum culling and hidden region removal

Ray casting, being an image-order front-to-back approach, is an effective visualisation technique appropriate for large systems, and intrinsically employs frustum culling and early ray termination (see Sec. 4.2).

Moreover, ray tracing is flexible and accurate since it operates with rays and not with the volumetric data directly. However, since volume data and rays are generally uncorrelated, *e.g.* the data access pattern over the volume is highly irregular, ray casting tends to suffer from memory incoherency, and thus poor cache exploitation and performance (for a detailed discussion, see for instance [193]). Therefore, ray tracing should be enhanced with cache-aware optimisation; below, we discuss a new data format to increase memory coherency and performance.

Transparent region skipping

The ability to quickly traverse transparent regions represents a fundamental ingredient in efficient volume rendering. In ray casting *transparent region skipping* is usually achieved by employing a hierarchical min/max acceleration structure, *e.g.* a

min/max kd-tree [196] (as mentioned in Sec. 4.2). Here, each node in the tree encapsulates the minimum and maximum values of their daughters and each tree leaf stores the minimum and maximum values of the spatially spanned voxels; ray traversal entails starting from the root node and recursively proceeds in a front-to-back order, examining smaller nodes only if they satisfy certain criteria. For example, for isosurface rendering, a node is skipped if the isovalue is outside the min/max range stored in that node; for volume rendering a node may be considered transparent if the difference between the stored maximum and minimum is lower than a user-defined threshold, as described in [210].

Acceleration structure for effective hidden region removal

With respect to time-varying volume rendering, the main bottleneck pertaining to min/max trees and other more involved methods for handling transparent region skipping is that they inherently depend on volume data; thus, if the latter changes the former must be re-built, which offsets the benefits related to ray casting (described above) and prevents interactive visualisation. Moreover, the employment of optimised parallel min/max tree construction is not sufficient for large systems [210]. We circumvent this problem in the manner discussed below.

Generally the non-transparent part is sparse and occupies a small fraction of the volume bounding box. We enhance the benefit offered by sparsity by means of the following method. During the preprocessing stage, the volume of interest is subdivided into a set of disjoint subdomains by means of a simple graph growing partitioning (GGP) algorithm. Each subdomain is assigned the same number of voxels and generally comprises one or more disconnected clusters of voxels ('volume clusters'). The GGP technique grows N subdomains from N corresponding seed points/voxels by sequentially applying a breadth-first expansion algorithm to each seed point (see Chapter 3 for a complete description). During volume rendering of each time frame ray casting is applied to each volume cluster independently, considering only the rays sent from the eye through every pixel spanned by the minimum viewport rectangle containing the projected vertices of that cluster. Therefore, a set of sub-images is produced, one for each volume cluster. All the sub-images are correctly assembled to form the complete image; merging is aided through a look-up table. Specifically, a z -buffer is used for isosurface ray casting and the associativity rule is exploited during volume rendering [231]. Several advantages characterise this ray casting approach as explained below; we also discuss the potential disadvantages and some techniques to circumvent them.

Ray traversal is usually regarded as the most expensive aspect of ray tracing, particularly in large systems [196]. Splitting the volume into a set of clusters, which closely adhere to non-transparent regions, and ray casting them independently, holds

the potential to drastically reduce the number of rays to cast, ray-box intersections and traversal steps, as shown in Fig 4.1.

However, multiple rays are cast through a pixel if more than one cluster maps to that pixel. This occurs when a particular view point results in two or more screen-cluster projections overlapping. This leads to undesirable ray initialisations and traversal steps following the first-hit voxels. One cannot avoid this situation for viewpoints generally and, as a consequence, our method has a certain overhead which may overwhelm its original benefits for a highly occluded cluster distribution. To avoid this we could use an image-based occlusion map: we first quickly project each cluster and determine their visibility in ascending order; secondly, for each pixel we consider the cluster nearest to the image-plane and we do not proceed to the next cluster if the ray is blocked.

If the bounding boxes of the clusters are not spatially disjointed some voxels may be pierced multiple times *i.e.* by rays pertaining to different clusters. A partitioning algorithm which is different from ours may be adopted to produce spatially-independent axis-aligned subdomains. However, if the spatial distribution of clusters is very sparse, the probability that a ray through a pixel spawns a non-transparent region, and therefore multiple clusters, is very low.

A shortcoming inherent in our approach consists in not supporting advanced shading effects which are based on secondary rays, *e.g.* shadows, as long as they capitalise upon rapid data access to any part of the global system information. However, this feature is not precluded since one can maintain a global ray traversal acceleration structure for secondary rays. Alternatively, the volume clusters themselves may be embedded in a memory cheap hierarchical acceleration structure which, as a consequence, encapsulates global information in the form of a spatial collection of local clusters.

Optimisation of ray traversal steps, memory coherence and data updating

Our volume rendering method resembles object-order ray tracing algorithms [192,232]. In contrast to these our basic algorithm does not rely on global ray traversal and data structures; this is the key aspect which significantly reduces the number of operations (ray traversal steps) and enhances memory coherency.

When a ray traverses a large volume, it samples non-contiguous data that are likely to reside far from each other in memory address space, thus provoking numerous cache misses and poor performance. The volume data associated with a cluster is stored in a two-level grid, namely a regular grid of macrocells (or bricks) each of which is a rectilinear grid of voxels. Therefore, ray traversal is completely shifted from a global approach over a large dataset to a cache-friendly local one. The multi-level representation further improves memory coherence [197]. Furthermore, our compact two-level

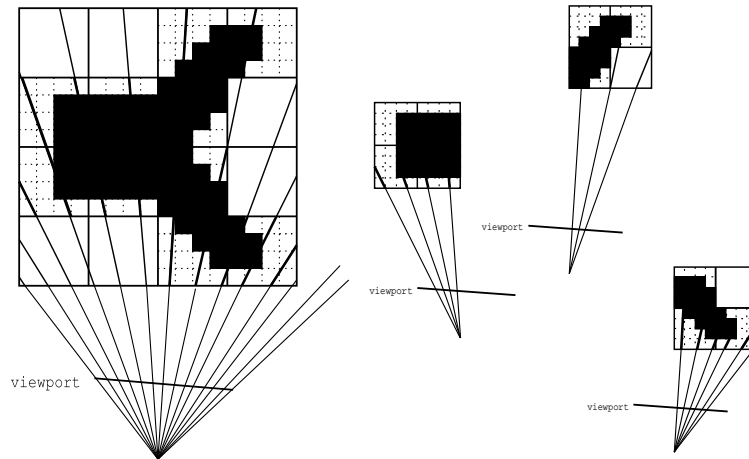


Figure 4.1: Two-dimensional representation of a ray casting approach to handle volume rendering of a voxelised bifurcating 3D dataset; (a) standard technique (left image) and (b) our method (three rightmost images) which decomposes this example system into three volume clusters (see Sec. 4.3.1 for details). The transparent and opaque voxels of the two-level grid used to accelerate ray traversal are represented with dashed borderlines and black squares respectively. Non-empty macrocells comprise 4×4 voxels. The computational cost of a ray is proportional to its thickness: a ray that is only generated and is tested against a bounding box for intersection purposes has the minimum thickness, while a ray which also traverses empty and/or full macrocells is thicker. The total number of ray generations and ray-box intersections is $4 + 3 + 6 = 13$ with our method (instead of 16); here, ray traversal within some macrocells and voxels is avoided because the bounding boxes of the subdomains are shrunk with respect to the original system (thus empty macrocells are not taken into account). Conversely, our method considers post-first-hit ray traversal steps which can be avoided in the conventional technique: our algorithm needs to assemble sub-images pertaining to different subdomains in a single image.

data structure incorporates an essential feature related to the pre-computation stage needed to accelerate transparent region skipping, which we explain below.

Min/max tree construction is accomplished by propagating the minimum and maximum information from the voxels up to the node root through inner nodes, following a bottom-to-top approach, as described in [210]. This time-consuming multi-pass algorithm optimises the ray traversal stage of static volumetric data in the presence of empty regions but simultaneously prevents rapid visualisation of time-varying volume data since it must be applied at every time frame. Fortunately, if we employ a two-level data format, we have to update the finest grid only, which involves a single step procedure. Further details are provided in Sec. 4.3.3 whilst discussing the coupling between the fluid solver and the renderer. Additionally, we determine concurrently whether a cluster of macrocells are completely transparent; in that case their computation during ray casting is completely avoided.

Colour mapping

We have chosen to employ a zero-order scheme for volume rendering. Specifically, for isosurface ray casting the ray is terminated at the nearest face of the first-hit voxel with flow field value greater than the chosen isovalue; a colour-code maps the voxel value to the pixel colour. During volume rendering, instead we calculate its integral by considering each non-transparent voxel pierced by the ray; each contribution is the colour associated with the voxel value multiplied by the segment pierced by the ray.

Zero-order approaches are fast but not very accurate. In fact, they can easily produce artifacts like patchy and aliasing effects. However, if the volumetric dataset is of a high resolution compared to the image display resolution, the visual accuracy does not depend on the sampling interpolants but rather on the spatial resolution of the simulated flow field. Furthermore, our volume rendering approach represents the fastest way to scan all the voxels pierced by the ray, in contrast to most of the ray casting techniques which sample the volume at small regular intervals. Additionally, the use of high order extrapolation methods may be not suitable because our flow fields are generally sparsely defined. Finally, sampling approaches whose order is not zero require knowledge of data that pertain to adjacent subdomains which are not provided by our fluid solver HemeLB; as a result, an extra communication stage should supply this information, which entails complicating the implementation, allocating further buffers and results in a less efficient strategy.

Optimised ray traversal engine

The voxel traversal technique developed by Amanatides and Woo [233] is employed to rapidly determine the voxels pierced by the ray in front-to-back order. The two-level clusters are explicitly and recursively traversed by using several optimisations

as follows.

Ray traversal is handled by one of eight routines, depending on the x -, y - and z -signs of the ray; this minimises the number of conditionals and saves computational operations. Moreover, ray traversal at the macrocell and voxel levels are approached by two different functions, which avoids further conditionals and ray-initialisation-steps³. Additionally, the fact that several operations can be avoided is exploited since (a) some parameters associated with the ray traversal at the finest level are not necessary if the ray is stopped at the first voxel, (b) macrocell spatial extent, needed to define the voxel of the finest level located by the current ray sample, can be computed incrementally and (c) integer multiplications needed to refer the element of the currently traversed grid, n , are minimised if one encodes in terms of the (iMN, jN, k) triplet instead of the (i, j, k) one, where $n = (iMN + jN + k)$ and the grid has (L, M, N) cells. Furthermore, we have augmented several ray initialisation and traversal steps with SIMD friendly code which can be easily vectorised on several architectures. In Sec. 4.6 a pseudocode implementation shows various optimisations of our ray traversal algorithm with respect to a naïve implementation.

The ray enters and leaves the minimum bounding box containing non-void voxels of the current cluster; this eliminates further traversal steps. Moreover, the viewport-projection of each cluster is very compact. The vertices of each minimum bounding box are pre-calculated. This optimisation which we have adopted doubles the speed of the ray tracing overall. The resulting code is quite simple and compact; higher-level hierarchies are not employed and complex pre-computation, such as caching [193], is not performed (see the discussion in Sec. 4.2).

4.3.2 Further optimisations

In the following sections we discuss weaknesses and strengths of our ray casting approach, and describe further improvements to enhance its quality and speed, typical of modern ray tracing systems.

Handling advanced rendering effects

We will see in Sec. 4.4.1 that our ray caster is very efficient and can render sparse large-scale three-dimensional volumes at high image resolutions at tens of frames per second on a single processor. Rendering quality may be easily enhanced through the incorporation of trilinear interpolation and normal estimation by considering neighbouring voxels adjacent to the ray samples. As we discussed in Sec. 4.3.1, this requires special care at boundary voxels, for instance, through some extrapolation techniques.

³In a traditional recursive approach which handles several hierarchical levels this is more difficult to control and involves the use of several conditionals and computations which may significantly degrade performance. Our bi-level ray traversal overcomes these problems.

Unfortunately, it is difficult to implement advanced rendering effects based on secondary rays, *e.g.* shadowing and global illumination, as in [196,206]; below, we explain the reason and provide some ideas to circumvent the problem.

The object-based nature of our ray casting approach prevents straight forward management of secondary rays. For a regular parallel domain decomposition the maximum number of neighbouring partitions is 26; the irregular domain decomposition into a set of sparse subdomains prevents to limit *a priori* the subdomains from which a ray may come from, which can be very large in number. In a shared memory system, MPI communications can be substituted by appropriately protected shared memory access operations, but our goal is to be able to render the flow field on any platform. The problem related to the MPI communication of secondary ray data does not hold when the processor count is small but might be a bottleneck when this is large. One way to circumvent the problem is to partition the empty space between the volume clusters by generating extra (empty) partitions such that each processor has a compact set of neighbouring ones. Unfortunately, this technique slightly offsets the advantage provided by rapid transparent region skipping which characterises our ray casting method since each empty partition must exchange ray data with neighbouring ones. One technique to render shadows adopts the shadow maps typical of software based on polygon rasterisation. First, the flow field is rendered from the light positions; for each light the corresponding image is stored in a buffer and eventually filtered; second, the flow field is rendered from the eye location which uses those buffers to determine whether the current pixel is in shadow or not. In this procedure, a projection from the three-dimensional location to the two-dimensional light buffer is employed.

Culling of further ray traversal steps

In this subsection, we discuss optimisation techniques to further eliminate ray traversal steps. As mentioned in Sec. 4.4.1, if each ray traverses several volume clusters our object-based ray caster might result inefficient because some traversal steps are performed for each partition, while the advantage of typical ray tracers is to stop at the first hit. We can obviate this problem by projecting all the volume clusters prior to the ray traversal stage; in this way we know how many of them are seen from each pixel. Thus, we can use a standard ray tracing approach if their number is above a certain threshold. In this hybrid technique we have the drawback of some memory overhead due to the global ray traversal, but this is eventually alleviated by re-using and linking the sparse representation of the novel data structure to a coarse and global hierarchical grid.

We send rays through the minimum two-dimensional image rectangle containing the eight vertices of the volume cluster. This often leads to unnecessary ray-box

intersection tests. It can be avoided by scan converting (rasterising) the volume box, which results in a compact representation of the pixels seen through this three-dimensional projection.

Further exploitation of underlying hardware

Data structures may be aware of the underlying hardware architecture, which is relevant to well exploit cache hierarchy and maximise performance [197]. In this work however, we did not study the influence of the size of the macrocells on the ray casting performance since we prefer to produce a platform independent solution.

Optimisation of spatial partitioning

Our ray casting performance is a function of the occlusion of the scene and the sparsity of the volume to be rendered: a poorly occluded system with substantial empty space is quickly rendered (see Sec. 4.3.1 and 4.4.1). We did not have further insight into this topic, but show a performance plot as a function of the number of subdomains for three different systems in Sec. 4.4.1. With some quantitative results or theoretical arguments one can optimise the number of subdomains and volume clusters for the current computational domain. Furthermore, the partitioning into subdomains can be recursively performed and optimally selected by checking, at each level, the current extent of occlusion and distribution of non-empty regions.

Enhancing hidden volume removal

One disadvantage of object-based rendering approaches is that some processing pertaining to volumes outside the view frustum, *i.e.* of those which are culled, cannot be avoided, given that substantial computation is dedicated to the other regions. Moreover, in a parallel implementation this has a highly undesirable consequence: processors dealing with culled volume clusters are idle while the others are responsible for the majority of the computation. Thus, a significant load imbalance arises when the user zooms into certain locations of the scene. One way to mitigate the problem is to switch from a ray casting approach to a raster-based one to accelerate the rendering of the visible parts for which the ray casting would be quite slow. Thus, several improvements can be employed to enhance performance and quality, but nonetheless our ray caster is very efficient (as shown in Sec. 4.4.1) and its simplicity is the key of its usability.

4.3.3 *In situ* visualisation

In Sec. 4.2, we discussed some of the features of *in situ* visualisation techniques, specifically coupling the lattice-Boltzmann fluid solver HemeLB and the ray casting

technique, as presented in Sec. 4.3.1. Here, we describe our cheap cache-friendly data format of each volume cluster and the communication pattern that permits us to assemble the sub-images corresponding to different MPI ranks.

At the beginning of the simulation, a subdomain is assigned to each MPI rank, each subdomain being represented by a set of clusters (see Sec. 4.3.1). Local fluid simulation data on each MPI rank are based on reordered one-dimensional arrays for efficient purposes (see Chapter 3 for further details). The mappings between the two-level volume clusters and the one-dimensional arrays are maintained through a two-level grid and ray casting is locally performed within each cluster (Sec. 4.3.1). Therefore, the pressure or velocity magnitude or the stress flow field is directly stored as a fragmented collection of two-level hierarchies, one for each cluster. The simulation and the rendering are performed in double and single precision respectively. When a voxel value is changed during the simulation we need to update the corresponding one in the proper cluster. Here, the key aspect is played by a one-dimensional array of pointers which are set once to map the one-dimensional simulation data to the fragmented flow field. If we want to update the min/max values of the cluster of macrocells to further cull traversal steps during ray casting (as discussed in Sections 4.2 and 4.3.1), we need to use an extra (short) array for each cluster sized as its number of macrocells.

These data structures permit us to admit only a small memory overhead and to directly and efficiently update the data whose layout is optimised for volume rendering. These are central aspects that distinguish our visualisation engine from many others which are afflicted by long pre-processing times and/or substantial memory overheads (see Sections 4.2 and 4.3.1).

4.3.4 The compositing scheme

The scheme developed by Stompel *et al.* [217] has been regarded as the most efficient compositing method. Unfortunately, it has a serious disadvantage for the volume rendering of sparse systems: it assumes that each subdomain is a parallelepiped and thus its corresponding sub-image is “full”, which is not our case due the sparsity of the spatial distribution of the non-transparent voxels. We have implemented a binary tree communication pattern, as shown in Fig. 4.2. At each tree level each subimage to be sent is larger than that at the previous level which favours point-to-point communication bandwidth. At the same time, more and more processors are idle and work load imbalance grows; however, this is inevitable consequence of storing the entire image on one processor.

In contrast to the algorithm presented in [217], a compressed representation of each sub-image is maintained: only coloured pixels are stored along with the identifier as well as (a) the pixel colour and the length of the ray route for volume rendering or (b)

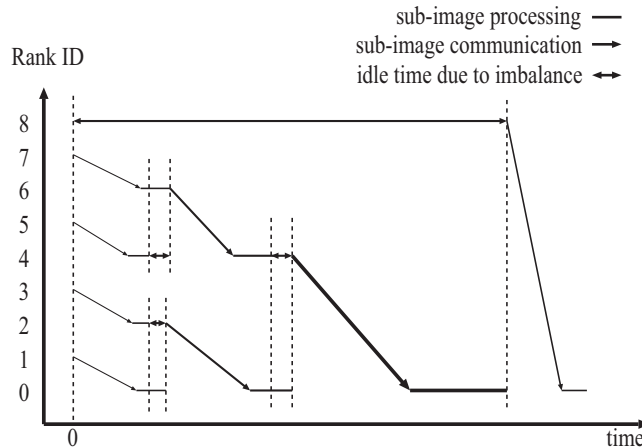


Figure 4.2: Illustration of our binary tree communication pattern over time to handle image compositing using 9 processors numbered 0 to 8. Each processor sends its sub-image to a neighbouring one with lower rank, specifically depending on the tree level. The thickness of each arrow denotes the message size, while the one of each line represents the time to merge the received sub-image. At the end of the process the 0-rank processor has the final image which can be written to file or communicated to a remote computer. Load imbalance can arise, which is a result of the binary tree communication pattern used as well as different sub-image sizes.

the first-hit voxel value and distance from the eye for isosurface ray casting. It is useful to display the simulation results in different modalities; therefore, we have chosen to render the current flow field by means of two isosurface-based and two volume-based visualisations (see Sec. 4.4), which are accomplished through a single-step-per-pixel ray traversal. The performance of this compositing scheme will be explored in Sec. 4.4.

4.3.5 Steering approach

Computational steering plays a crucial role in data exploration and scientific discovery [220, 221]. Some aspects concerning steering are discussed in Chapter 5 while the details of the parameters which can be steered and monitored are outlined in Chapter 6. Here, we describe a steering environment to monitor simulation results as well as interact with the fluid solver and the ray caster to modify simulation and rendering parameters at interactive frame rates.

Our novel strategy to handle *run-time* steering and visualisation involves minimising the need for high-bandwidth communication between different components and software applications of our computational infrastructure - the renderer, fluid solver, supercomputer and remote computer, and the coupling between steering and simulation/rendering (see Fig. 4.3). Outgoing composed images and incoming steering signals are handled through two threads running on rank 0, and domain decomposition is performed in such a way to assign the rank 0 no lattice sites to avoid overload.

Thus, the main task of rank 0 is to collect the composed frame as well as broadcast necessary steering parameters to all other ranks. By using asynchronous threads in this fashion, network bandwidth limitations, client disconnects, etc. do not affect the speed of the underlying simulation. Frames are only rendered and composed when no frame is being transmitted over the network to the client (i.e. as they are needed) (Fig. 4.3). In this way, in contrast to other interactive environments, such as that presented by Belleman and Shulakov [223], two important features arise and confer our method a certain strength: (a) rendering is performed at a high maximum interactive frequency, only limited by the network bandwidth between the simulation and steering/visualisation client, and (b) due to the asynchronous network transmission, rendered image transfer does not affect the underlying simulation speed.

Compressed images are faster to be transferred; unfortunately, the time needed to compress them may be much longer than the rendering time. Therefore, we do not apply any compression scheme but we avoid the storage and network transmission of non-coloured pixels by only sending the coloured ones.

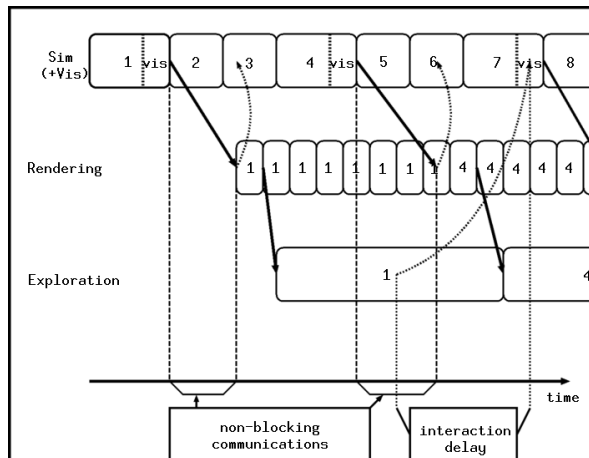


Figure 4.3: Temporal sketch of our interactive simulation environment. Depending on the pipeline stage from top to bottom, the number in each box refers to the simulation time step currently processed, rendered or displayed, and finally explored. The simulation and volume rendering take place on the same platform, while the image of the flow field is transferred to a remote computer client and displayed. The clinician can explore and interact with the simulation output by interactively using the steering capabilities. One advantage of the scheme used is that the fluid solver is not substantially affected by the speed of the other environment components (see Sec. 4.3.5 and 4.4 for further details and performance results respectively).

4.4 Performance results

In this section, we present an extensive overview of performance results of our visualisation engine. We first report the single-processor performance of our ray caster bi-level algorithm on time-independent fluid flows and compare it to one based on an explicit kd-tree technique (Sec. 4.4.1). The conventional kd-tree-based ray tracing algorithm relies on one kd-tree. Here, instead, we also monitor the performance of our object-based subdivision technique (Sec. 4.3.1) by representing each volume cluster by a separate kd-tree.

In Sec. 4.4.2 parallel performance (intrinsic speed and scalability) of the *in situ* visualisation engine incorporated in HemeLB (Chapter 3) is discussed in the context of our ray-caster and compositing approaches, for the simulation of time-varying flow fields. In Sec. 4.4.3 performance of the remote interactive and steering capabilities is discussed.

4.4.1 Ray casting performance

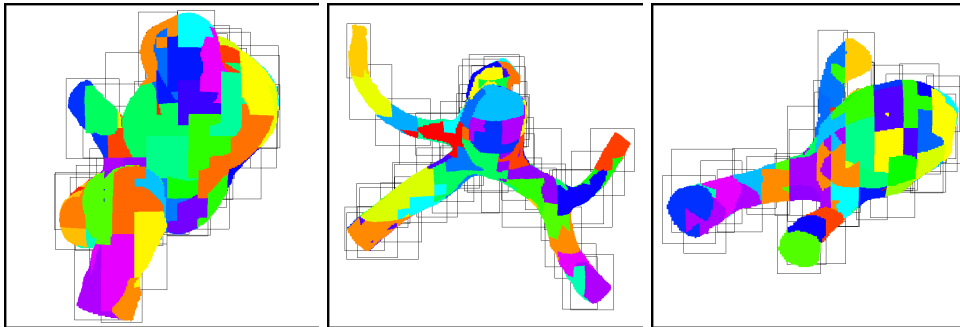


Figure 4.4: From left to right, we show the first, second and third models considered in this work (referred to as Model 1, Model 2 and Model 3). They correspond to three highly-resolved digitally reconstructed patient-specific intracranial vascular pathologies and comprise 1661545, 183228 and 249094 sparsely distributed fluid voxels respectively. In the three images, we depict 64 colour-coded subdomains. Each subdomain is subdivided into one or more volume clusters which are rendered independently by tracing rays through their minimal viewport rectangles (see Sec 4.3.1 for details). The boxes are outlined in black.

Here, we compare the single multi-core processor performance of our ray caster to one relying on an explicit kd-tree [196, 206], which is the state-of-the-art of software-based isosurface visualisation of static flow fields (Sec. 4.2).

For the explicit kd-tree ray caster, as done in our object-based ray casting technique, the whole system is decomposed into a set of subdomains. Each volume cluster

is represented by a two-level grid or an explicit kd-tree and ray traced independently. Here, the domain decomposition is performed only once and during dynamically evolving *in situ* blood-flow simulations most of the data structures have already been constructed by HemeLB; thus no pre-computation and replication of buffers is required (Sec. 4.3.3). All benchmarks were performed by having $8 \times 8 \times 8$ voxels for each macrocell and by not including the time to display the coloured pixels. Inter-processor communications and hence the parallel compositing algorithm presented in Sec. 4.3.3 were not needed here; the N sub-images were assembled by compositing the image buffer of thread 0 with the remaining $N - 1$ threads.

The single-processor performance is evaluated by rendering the fluid flow confined in three digitally reconstructed patient-specific aneurysms. The original medical datasets comprise $512 \times 512 \times 400$ voxels but we increased this resolution five times along each axis for simulation accuracy purposes. However, the region of interest of each model is much smaller than $2560 \times 2560 \times 2000$ voxels and the fluid voxels, *i.e.* the ones processed by the fluid solver, are sparsely distributed within compact bounding boxes; there are 1661545, 183228 and 249094 fluid voxels for the first, second and third model respectively; see Fig. 4.4 where they are subdivided into 64 subdomains. For the remainder of this chapter we refer to these models as Model 1, Model 2 and Model 3 respectively. Notably, these systems differ in terms of the number of fluid voxels and in structural properties which diversify the level of visual occlusion (see Chapter 6 for further details on these aneurysm models).

We use the ray caster to render the first-hit pierced voxels that have a flow field value higher than a specified threshold. This allows us to render the pressure or the stress at the outermost voxel (artery) layer, which is clinically important. The conventional isovalue-based approach requires storage of the maximum flow field value in the current volume cluster, macrocell or kd-tree node, as opposed to the min/max counterparts which record either their minimum or the maximum values. We call these incomplete min/max objects “zero/max” ones. This methodology is less restrictive than a typical isovalue-based ray tracing which only considers the surface of voxels adjacent to the current isovalue, and therefore is penalised because it forces us to consider volume clusters or macrocells that do not contain the isovalue and are skipped in a traditional isosurface technique.

The performance results in terms of frames per second (FPS) for a viewport of 800^2 pixels achieved by using a single Intel Core 2 Quad 2.5 GHz and the Intel 9.1 compiler with flags `-O3`, `-ipo`, `-xT` are reported in Figures 4.5, 4.6 and 4.7. We compare the performance attained by the novel object-based zero/max two-level grid ray caster to our object-based zero/max explicit kd-tree one. The whole system is decomposed into a set of subdomains and each of their volume clusters is represented by a kd-tree (or a bi-level grid), which is ray traced independently (see Fig. 4.4). The latter becomes a

standard kd-tree ray caster running on a single processor core if the spatial partition comprises a volume cluster; as a consequence, we check either its performance or test the efficacy of the nature of our object-based technique. The structure of the kd-tree node requires 128 bits and thus fits a cache line, which enhances performance. Shared-memory parallel rendering of the subdomains is achieved by a relatively simple OpenMP implementation, where the number of OpenMP threads used is the minimum of four and the number of subdomains. Notably, the compositing stage is much simpler than that presented in Sec. 4.3 since sub-frame buffers are merged with memory copy operations.

We note that the single-core single-subdomain performance of the ray tracer based on the bi-level grid is significantly higher than the one relying on the explicit kd-tree. Notably, we did not include the computational time due the zero/max kd-tree construction which is about 0.07 seconds when applied to the largest vasculature (Fig. 4.5), and therefore comparable to the rendering time. Conversely, the updating cost of the zero/max bi-level grid is insignificant and, if employed in combination with a fluid solver through an *in situ* approach (as discussed in Sections 4.3.3 and 4.4.2), can be performed with no intermediate processing tasks.

For all vasculatures, the single-core performance of the bi-level-based ray caster becomes dramatically superior by splitting the vasculature into approximately 8 subdomains, which demonstrates the effectiveness of our partition-based visualisation approach. The employment of OpenMP is also particularly beneficial; the ray caster augmented with the bi-level scheme together with the partitioning sustains excellent frame rates. Additionally, the performance obtained by visualising the isosurface of the velocity flow field is similar to or better than that achieved by stopping the ray casting at the first fluid voxel as carried out during the rendering of the pressure distributed at the aneurysm wall.

4.4.2 Parallel ray casting performance and scalability

Here we report on parallel scalability and performance of the *in situ* visualisation approach. For each vasculature four different visual modalities were simultaneously rendered at 512^2 resolution. Snapshots of the output of the three datasets used are shown in Fig. 4.8. The Ranger supercomputer at the Texas Advanced Computing Centre (62,976 AMD Opteron Barcelona processors with InfiniBand interconnects) was used to perform the runs with the MVAPICH MPI library v1.0.1 and Intel v10.1 C/C++ compiler with flags `-O3 -xT -ipo`. As carried out in the single-processor performance scenario, each macrocell was $8 \times 8 \times 8$ voxels in size.

The performance is shown in Fig. 4.9 as strong scaling, the problem size (number of fluid sites) being kept constant as the number of processors is increased. Each rank in the MPI communication space has its own subdomain and on each rank these

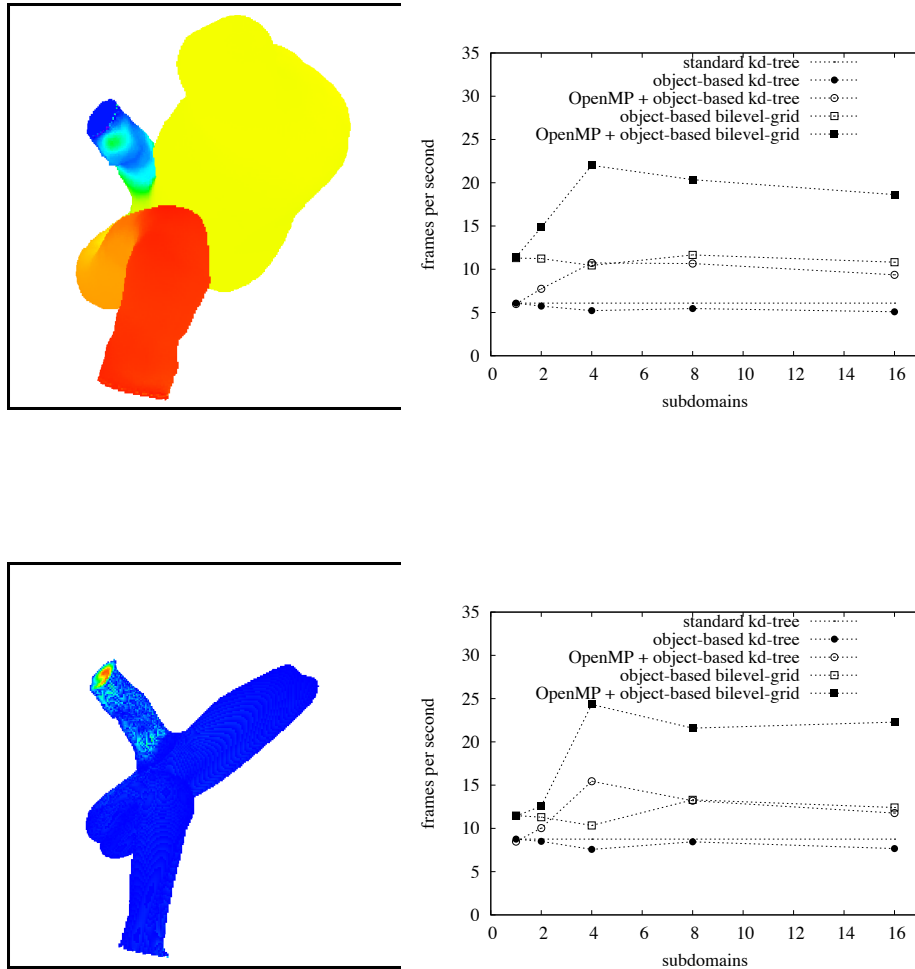


Figure 4.5: The top-left image shows the pressure distribution at the wall of Model 1, a large aneurysm comprised of 1661545 fluid voxels, while the bottom-left image depicts the part of the vasculature characterised by a velocity greater than 20% of the maximum value. The colour scale depicts low (violet) values to high (red) values. The plots on the right hand side report the performance achieved on a single Intel quad-core 2.5 GHz with a viewport of 800^2 pixels using various volume rendering approaches as a function of the number of partitions in which every vasculature is subdivided. Specifically, the horizontal dashed line is the single-core performance achieved by the explicit kd-tree. We applied the object-based technique presented in Sec. 4.3.1 to the ray caster based both on our bi-level grid and the explicit kd-tree; their performances are also shown. See Sec. 4.4.1 for further details.

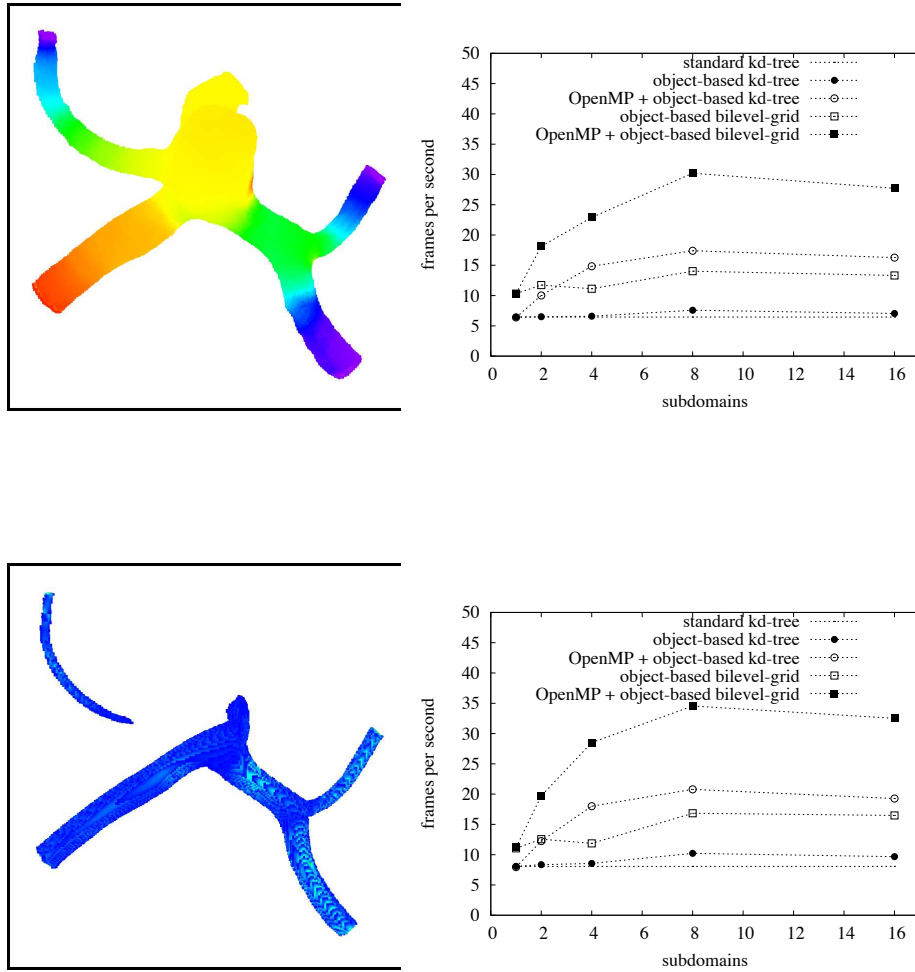


Figure 4.6: The top-left image shows the pressure distribution at the wall of Model 2, a small aneurysm comprised of 183228 fluid voxels, while the bottom-left image depicts the part of the vasculature characterised by a velocity greater than 20% of the maximum value. The colour scale depicts low (violet) values to high (red) values. The plots on the right hand side report the performance achieved on a single Intel quad-core 2.5 GHz with a viewport of 800^2 pixels using various volume rendering approaches as a function of the number of partitions in which every vasculature is subdivided. See Sec. 4.4.1 for further details.

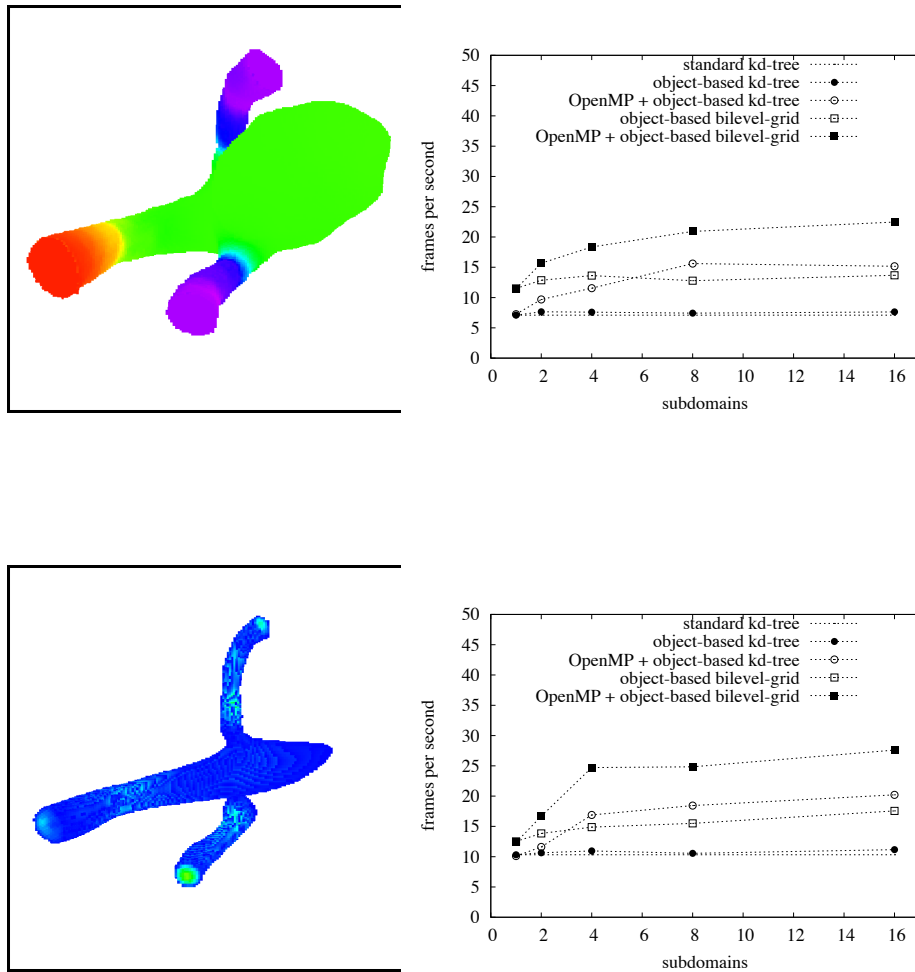


Figure 4.7: The top-left image shows the pressure distribution at the wall of Model 3, a small aneurysm comprised of 249094 fluid voxels, while the bottom-left image depicts the part of the vasculature characterised by a velocity greater than 20% of the maximum value. The colour scale depicts low (violet) values to high (red) values. The plots on the right hand side report the performance achieved on a single Intel quad-core 2.5 GHz with a viewport of 800^2 pixels using various volume rendering approaches as a function of the number of partitions in which every vasculature is subdivided. See Sec. 4.4.1 for further details.

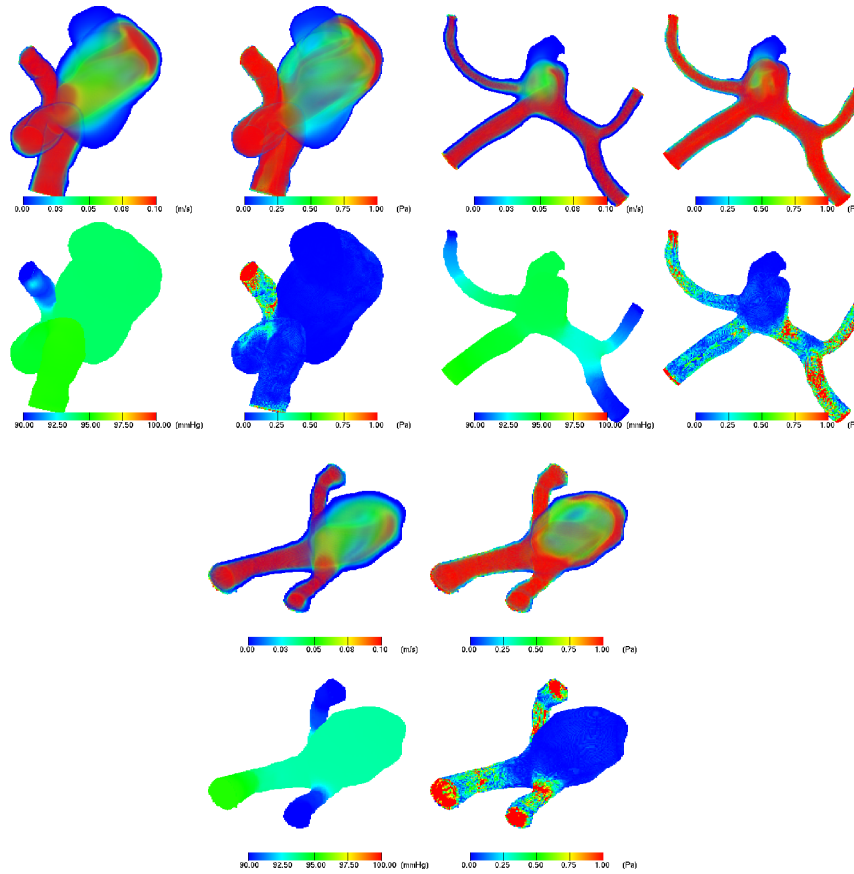


Figure 4.8: Snapshots of the simulation results obtained with our *in situ* visualisation system. The three different sets of four images correspond to three different types of vascular pathologies which are Models 1 (top-left set), 2 (top-right set) and 3 (bottom set). For each model, the volume rendering of the velocity and stress flow fields of the flow half way through a pulsatile cycle are depicted on the top-left and top-right images respectively. The wall pressure and wall stresses are shown in the bottom-left and bottom-right images respectively.

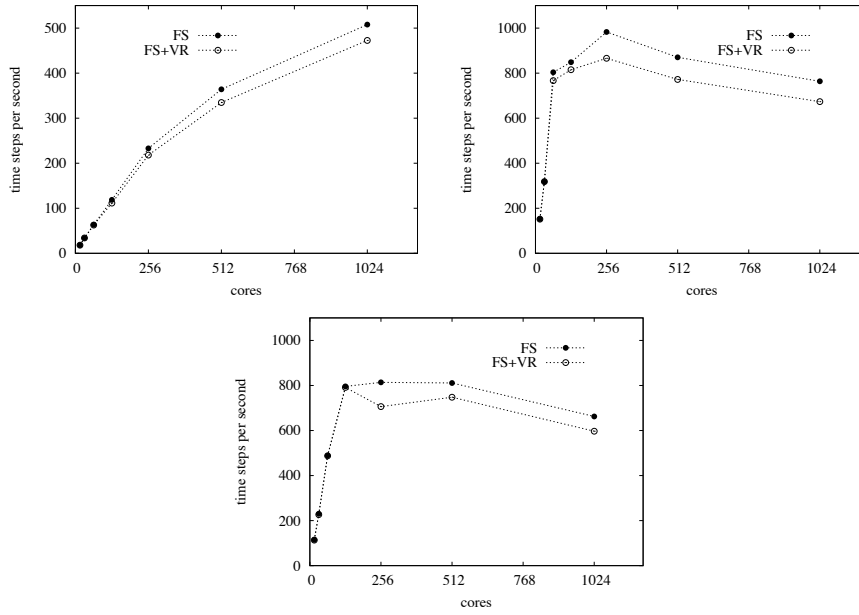


Figure 4.9: Parallel scalability and performance of the *in situ* rendering system for the vascular pathologies shown in Fig. 4.8, namely Model 1, 2 and 3. The flow field is simulated using the parallel fluid solver HemeLB and the volume rendering is performed by the *in situ* object-based ray caster built into HemeLB (Sec. 4.3.1). Performance of the fluid solver alone (FS) and the fluid solver with volume rendering (FS+VR) are shown as time steps per second.

subdomains are split into multiple volume clusters. The scalability of the fluid solver alone as well as use of *in situ* rendering at every time step is particularly clear in the first dataset with 1661545 fluid sites (≈ 1600 sites per rank using 1024 cores). Once the number of fluid sites per rank drops below 1500 – 3000 scalability is reduced because of the heavy increase in communication relative to computation on each MPI rank. Of course this depends on many factors associated with the machine, such as the MPI communication fabric and the local processor speed. It is worth noting that the scalability of the fluid solver plus ray tracer is superlinear for 128 to 256 cores for Model 1 and 16 to 64 cores for Models 2 and 3.

High LB simulation rates (up to 0.831×10^9 fluid site updates per second) are achieved even with volume rendering taking place at every time step. Most pulsatile hemodynamic simulations are run using 25×10^3 to 100×10^3 time steps per pulsatile cycle, so practically speaking we can simulate and visualise a pulse in less than 60 seconds in a typical dataset. The impact of the volume rendering is less than 10%; however, in practice volume rendering does not take place at every time step since, during an interactive session the flow field does not need to be rendered at more than 25 frames per second (FPS). This is discussed in the following section.

4.4.3 Interactive performance

Table 4.1: Performance results for interactive steering with the display outputs for the three datasets used; their snapshots are shown in Fig. 4.8 when the image size is large.

Dataset	#cores	Image size (zoom)	FPS	LB steps/s	LB steps/s wo Steer/Vis
Model 1	1024	tiny (0.5)	15.2 ± 1.9	170 ± 26	323
		large (20)	1.5 ± 0.5	170 ± 75	
Model 2	128	tiny (0.5)	16.2 ± 0.5	641 ± 23	670
		large (25)	0.5 ± 0.1	626 ± 52	
Model 3	128	tiny (0.5)	8.6 ± 0.5	306 ± 40	653
		large (28)	3.5 ± 0.5	275 ± 51	

The key enabling capability realised by using this integrated rendering approach is to be able to visualise fluid flow as the underlying LB simulation is taking place. The near-optimal number of cores in terms of scaling to run on for the three models were 1024, 128 and 128 cores respectively (Fig. 4.9). Simulations were run within advance reservations on the Ranger cluster at the Texas Advance Computing Centre located in Austin, Texas on the US TeraGrid, manually reserved by local systems personnel. On other infrastructure which supports it we use HARC, the Highly Available Resource Co-allocator, for user-set advance reservations [234]. Transmission of the steering and rendered frames took place over a shared network infrastructure to/from UCL in London. Typical results are shown in Table 4.1. The frame rate is heavily dependent on the (fluctuating) network bandwidth, with large rendered frames in the order of 1.5 MB. The LB time-step rates are slightly lower than those plotted in Fig. 4.9 because we run in the full-production mode, instead of the benchmarking one; therefore, here the timing results include the computation of several blood flow parameters (see Chapter 6 for details).

The impact of steering and network transmission is evident from the reduction of the underlying LB speed as compared to the results in Fig. 4.9. For example, peak performance of Model 3 is of the order of 650 steps/s and is reduced to 306 ± 40 step/s during the interactive session. This is due to the two threads attached to rank 0 which deal with network transmission of rendered frames and handling of incoming steering signals; they are in contention for CPU resources with the underlying LB execution. Although rank 0 has no lattice sites associated with it, it still takes part in global MPI communications and is synchronous with respect to the rest of the MPI communication space; therefore any decrease in performance on rank 0 affects other

ranks.

Even so, the underlying LB simulation is executed at a high enough rate that hemodynamic features such as the change in the stress at the vessel walls can be observed interactively; the evolution of a pulsatile cycle of blood flowing through an aneurysm can be observed by a clinician within a few minutes. Even over international links, the network delay is minimal and delays in changes of the rendered images due to steering rotation, zoom, colourmap editing, etc., are rarely observed.

We have presented a visualisation pipeline to efficiently render and interactively explore time-varying flow fields. The main parts of the visualisation pipeline are the ray casting engine, the interactive steering capabilities and *in situ* rendering approach.

An essential requirement for maintaining optimal levels of performance is to keep the computational impact and memory required by the rendering and interactive kernels low with respect to the fluid solver. The object-based ray casting approach built on a bi-level grid is faster than an explicit kd-tree approach, which is the state-of-the-art of software isosurface rendering static datasets. The parallel implementation presented here is based on MPI and as a result our software is highly portable.

To the best of our knowledge no other technique for the visualisation of highly-resolved time-varying flow fields matches the performance reported here. Steering has been incorporated as a means to adjust rendering and simulation parameters interactively, drastically reducing turnaround time for result generation. The approaches presented here are quite simple and straightforward to implement; we encourage adoption of these concepts by other research groups involved in visual studies of fluid dynamics, particularly in the lattice-Boltzmann arena.

At the core of this approach is our aim of furnishing researchers and clinicians in our group with software to interactively examine blood flow through the human vasculature. The time from data acquisition to the delivery of visual output is minimised, making it ideal for clinical use where timeliness is of highest priority.

4.5 Conclusions

We have presented a visualisation pipeline to efficiently render and to interactively explore time-varying flow fields. The main components of the visualisation pipeline are the ray casting engine, the interactive capabilities and *in situ* rendering architecture. The first two aspects present a number of novel features; every one is highly optimised and, consequently, the performance of the entire software application is very high. One of its essential characteristics is to keep the computational impact and memory footprint required by the rendering and interactive kernels low with respect to the fluid solver, which is highly desirable. The object-based ray casting approach relying on our bi-level grid is faster than that based on the explicit kd-tree, which is the state-

of-the-art of isosurface rendering of static datasets. The parallel implementations are based on MPI and, as a result, our software is highly portable and can be used in either shared memory systems or distributed ones. To the best of our knowledge, no other technique for the visualisation of highly-resolved time-varying flow fields matches our performance. Furthermore, the entire software package is extremely useful for us, since it is aimed at effectively investigating flow fields confined in cerebral vascular trees. Steering has been incorporated, which is very important as a means to play with rendering and simulation parameters in real time, to shorten turnaround and compute times. Finally, we emphasize that the new algorithms are quite simple and we wish to encourage their implementations and adoption by other research groups involved in other scientific activities.

We have already mentioned the weaknesses and strengths of our ray casting approach. For example, it is difficult to fully exploit occlusion culling and to implement complex rendering effects like shadows and global illumination through inter-reflections. However, our visualisation system is very fast for time-varying sparse volumetric datasets by leveraging on compact memory-coherent data structures and optimised ray traversal. Furthermore, our *in situ* rendering core and interactive capabilities are cheap in terms of computational resource requirements, robust and fast for our current purposes.

4.6 Appendix: Naïve versus optimised ray traversal engine

Here, we provide pseudocodes for the naïve and optimised ray generation and traversal engines (Sec. 4.3.1). The core traversal algorithm is that presented by Amantides and Woo [233]. Entry point location and ray data initialisation needed for voxel traversal are always optimised by setting the voxel dimension equal to one (therefore multiplications or divisions by the voxel dimension are not needed). Below, the voxel traversal is suitable for isosurface rendering, while the algorithmic version optimised for volume rendering is slightly different. We assume that the x -, y - and z - signs of the ray $\mathbf{r}.*$ are 0, 1 and 1 respectively, *i.e.* $\mathbf{r.dir}[0] \leq 0$, $\mathbf{r.dir}[1] > 0$ and $\mathbf{r.dir}[2] > 0$; the other seven cases of macrocell (brick) and voxel traversal are handled in a similar way. $\mathbf{r.org}[i]$ is the ray origin and $\mathbf{r.inv}[i] = 1/\mathbf{r.dir}[i]$. $0..n$ indicates a loop for which the count runs from zero to n ; if $n = 3$ the corresponding loop can be easily vectorised by several compilers on various architectures. B_SIZE is the number of voxels along each side of every brick, $1 \ll B_SHIFT = B_SIZE$ and $B_SHIFT2 = 2 * B_SHIFT$. In contrast to the naïve algorithms, the optimised ones approach x, y, z computations by means of SIMD-friendly code. Most of the declarations are omitted for brevity while blank lines and bold text indicate the differences between the two versions.

Naïve ray generation

for each cluster c

```
calculate its rectangular projection;
perform clipping if needed;
for each pixel  $(i, j)$  therein
    calculate ray parameters  $r$ .*;
    ray versus cluster-box  $\rightarrow (t_{near}, t_{far})$ ;
    skip  $(i, j)$  if  $t_{near} > t_{far}$  or
    if another ray has been sent through
     $(i, j)$  and the registered intersection
    is closer to the viewpoint;

    if (TraverseBricks011( $t_{near}, r, cluster[c]$ ))
```

update pixel data;

Naïve brick traversal

```
int TraverseBricks011( $t, Ray\&r, Cluster\&c$ )
     $x[0..2] = r.org[0..2] + t * r.dir[0..2] - c.x[0..2]$ ;
     $i[0..2] = (int)(INV\_B\_SIZE * x[0..2])$ ;
     $i[0..2] = \max(0, \min(c.bricks[0..2] - 1, i[0..2]))$ ;
     $bx[0..2] = c.x[0..2] + (i[0..2] \ll B\_SHIFT)$ ;

     $b\_id = ((i[0] * B\_SIZE + i[1]) * B\_SIZE) + i[2]$ ;
    if brick[ $b\_id$ ] is not empty or culled

        if (TraverseVoxels011( $i, c.x, brick[b\_id], t, r$ ))
            return OPAQUE;
     $tmax[0] = (bx[0] + B\_SIZE - 1 - r.org[0]) * r.inv[0]$ ;
     $tmax[1] = (bx[1] + B\_SIZE - r.org[1]) * r.inv[1]$ ;
     $tmax[2] = (bx[2] + B\_SIZE - r.org[2]) * r.inv[2]$ ;
     $tdelta[0..2] = B\_SIZE * r.inv[0..2]$ ;

    while(1)
        if ( $tmax[0] < tmax[1]$ )
            if ( $tmax[0] < tmax[2]$ )
                 $i[0] = i[0] - 1$ ;
                if ( $i[0] < 0$ ) return TRANSPARENT;

                 $b\_id = ((i[0] * B\_SIZE + i[1]) * B\_SIZE) + i[2]$ ;
                if brick[ $b\_id$ ] is not empty or culled

                    if (TraverseVoxels011( $\dots, tmax[0], r$ ))
                        return OPAQUE;
                 $tmax[0] = tmax[0] - tdelta[0]$ ;
            else
                similar to above
        else
            if ( $tmax[1] < tmax[2]$ )
                similar to above
            else
                similar to above
    end while
```

Optimised ray generation

for each cluster c

```
/*  $cluster[c].x[0..3]$  are the coordinates of
the cluster-box vertices with lower values */
 $cx[0..3] = cluster[c].x[0..3] - viewpoint.x[0..3]$ ;
calculate its rectangular projection;
perform clipping if needed;
for each pixel  $(i, j)$  therein
    calculate ray parameters  $r$ .*;
    ray versus cluster-box  $\rightarrow (t_{near}, t_{far})$ ;
    skip  $(i, j)$  if  $t_{near} > t_{far}$  or
    if another ray has been sent through
     $(i, j)$  and the registered intersection
    is closer to the viewpoint;
     $r.org[0..3] = t_{near} * r.dir[0..3] - cx[0..3]$ ;
    if (TraverseBricks011( $r, cluster[c]$ ))
         $r.t_{min} = r.t_{min} + t_{near}$ ;
    update pixel data;
```

Optimised brick traversal

```
int TraverseBricks011( $Ray\&r, Cluster\&c$ )

     $i[0..3] = (int)(INV\_B\_SIZE * r.org[0..3])$ ;
     $i[0..2] = \max(0, \min(c.bricks[0..2] - 1, i[0..2]))$ ;
     $bx[0..3] = (i[0..3] \ll B\_SHIFT) - r.org[0..3]$ ;
     $i[0] = i[0] * c.bricks[1] * c.bricks[2]$ ;
     $i[1] = i[1] * c.bricks[2]$ ;
     $b\_id = i[0] + i[1] + i[2]$ ;
    if brick[ $b\_id$ ] is not empty or culled
         $x[0..3] = -bx[0..3]$ ;
        if (TraverseVoxels011( $bx, x, brick[b\_id], 0, r$ ))
            return OPAQUE;
     $tmax[0..3] = (bx[0..3] + B\_SIZE) * r.inv[0..3]$ ;

     $tdelta[0..3] = B\_SIZE * r.inv[0..3]$ ;
     $tmax[0] = tmax[0] - tdelta[0]$ ;
    while(1)
        if ( $tmax[0] < tmax[1]$ )
            if ( $tmax[0] < tmax[2]$ )
                 $i[0] = i[0] - c.bricks[1] * c.bricks[2]$ ;
                if ( $i[0] < 0$ ) return TRANSPARENT;
                 $bx[0] = bx[0] - B\_SIZE$ ;
                 $b\_id = i[0] + i[1] + i[2]$ ;
                if brick[ $b\_id$ ] is not empty or culled
                     $x[0..3] = tmax[0] * r.dir[0..3] - bx[0..3]$ ;
                    if (TraverseVoxels011( $\dots, tmax[0], r$ ))
                        return OPAQUE;
                 $tmax[0] = tmax[0] - tdelta[0]$ ;
            else
                similar to above
        else
            if ( $tmax[1] < tmax[2]$ )
                similar to above
            else
                similar to above
    end while
```

Naïve voxel traversal

```
int TraverseVoxels011(bi[], cx[], voxel[], t, Ray&r)
  bx[0..2] = cx[0..2] + (bi[0..2] << B_SHIFT);
  x[0..2] = r.org[0..2] + t * r.dir[0..2] - bx[0..2];
  i[0..2] = max(0, min(B_SIZE - 1, (int)x[0..2]));
  tmax[0] = (bx[0] + i[0] - r.org[0]) * r.inv[0];
  tmax[1] = (bx[1] + i[1] + 1 - r.org[1]) * r.inv[1];
  tmax[2] = (bx[2] + i[2] + 1 - r.org[2]) * r.inv[2];

  while(1)
    v_id = (i[0] << B_SHIFT) + i[1];
    v_id = ((v_id << B_SHIFT) + i[2]);
    if voxel[v_id] is opaque
      register voxel value(s);
      r.t_min = t;
      return OPAQUE;
    if (tmax[0] < tmax[1])
      if (tmax[0] < tmax[2])
        i[0] = i[0] - 1;
        if (i[0] < 0) return TRANSPARENT;
        t = tmax[0];
        tmax[0] = tmax[0] - r.inv[0];
      else
        similar to above
    else
      similar to above
  end while
```

Optimised voxel traversal

```
int TraverseVoxels011(bx[], x, voxel[], t, Ray&r)

  i[0..2] = max(0, min(B_SIZE - 1, (int)x[0..2]));
  tmax[0..3] = (bx[0..3] + i[0..3] + 1) * r.inv[0..3];

  tmax[0] = tmax[0] - r.inv[0];
  i[0] = i[0] << (B_SHIFT2);
  i[1] = i[1] << B_SHIFT;
  while(1)
    v_id = i[0] + i[1] + i[2];

    if voxel[v_id] is opaque
      register voxel value(s);
      r.t_min = t;
      return OPAQUE;
    if (tmax[0] < tmax[1])
      if (tmax[0] < tmax[2])
        i[0] = i[0] - SQUARE_B_SIZE;
        if (i[0] < 0) return TRANSPARENT;
        t = tmax[0];
        tmax[0] = tmax[0] - r.inv[0];
      else
        similar to above
    else
      similar to above
  end while
```

Notably, the optimised codes are organised in a way to drastically reduce the total number of operations of the most demanding ray traversal task (voxel traversal which is already speeded up by letting the voxel size equal to one), and to be further accelerated by SIMD-wise operations with vector length of 4.

Chapter 5

Haemodynamics on computational grids

Nothing changes your opinion as a friend so surely as success – yours or his

Franklin P. Jones

Fluid flow simulation in combination with patient-specific medical imaging data help to understand haemodynamic features of normal and malformed vasculatures and provides a tool which surgeons can use in real time to help plan courses of surgical treatment (Chapter 1).

Certainly, the fluid modelling of something as complex as the vascular structure of the brain requires large-scale computational resources; for real-time results, typical unsteady flow simulations require hundreds or thousands of processor cores. In order for patient-specific medical simulation to function in a real-world setting, not only is the correctness of the results important, but their timeliness is imperative to effectively support surgeons in making operative decisions. Various facets of high-performance computing are used to achieve this: (a) efficient blood flow modelling, (b) effective segmentation and manipulation of medical datasets, (c) *in situ* visualisation, (d) remote steering and visualisation, (e) distributed computing, (f) advance reservations and urgent computing capabilities, (g) automated job launching, (h) rapid data migration and (i) transparent access to resources.

This chapter focuses on how these various capabilities are implemented and exploited to provide a persistent, stable framework to aid courses of neurovascular surgery planning. The resulting problem solving environment is easy to use and very effective, and has played a key role in the studies presented in Chapter 6. The surgeons can use this middleware, not only to examine the pressure and velocity variations through the vasculature, but also to predict what changes might occur as a result of surgical intervention, which would be an invaluable addition to their surgical tool kit.

5.1 Overview

In this section, the individual software components which make up the problem solving environment are described. These were developed wholly by us, colleagues at the University College London, Manchester University and staff of the Louisiana Optical Network Initiative (see the acknowledgements at the end of the thesis). Specifically, Mazzeo developed the fluid solver HemeLB (Chapter 3), all its *in situ* visualisation capabilities (Chapter 4), the graphical-editing tool which pre-processes the data needed by HemeLB (see Chapter 6) and the rendering core of the HemeLB-GUI (see Sec. 5.2), Dr. Steven Manos and Stefan Zasada implemented the steering capabilities and the Application Hosting Environment respectively (see Chapter 4 and Sec. 5.2) while several people and institutions (cited in Sec. 5.2) contributed to the other aspects and kernels of the infrastructure *e.g.* the job reservation system and urgency computing mechanism.

5.1.1 Previous work

In Hassan *et al.* [47], an efficient modelling pipeline which includes data pre-processing, accurate CFD-based haemodynamic characterisation and visualisation of results was presented. Belleman and Shulakov [223] presented an efficient pipeline to handle visualisation and exploration of results while the simulation is running. Insley *et al.* [224] presented a similar interactive simulation system further enhanced by advanced grid capabilities. Sloot *et al.* [235–238] built a more complete and effective grid-based toolkit that incorporates several interoperating software packages to address haemodynamic studies. Specifically, their tools reconstruct and manipulate the medical dataset, distribute the computation on one of the 16 European sites involved in the CrossGrid project [239], handle the visualisation of the results, access the medical image repositories, deal with data migration between different software applications and monitor the results.

5.1.2 Present work

In the problem solving environment presented in this work, the parallel lattice-Boltzmann code HemeLB (see Chapter 3) is employed to effectively simulate fluid flow within complex intracranial vasculatures. Rather than relying on post-processing visualisation, an *in situ* volume rendering approach is used (see Chapter 4). Each rendered image is transmitted over the network to a lightweight client, resulting in immediate real-time visualisation of the blood flow field. The lightweight client permits one to steer HemeLB, where physical parameters of the vascular system along with various visualisation properties can be adjusted in real time. HemeLB is used in combination with MPIg, the latest release of the successor to MPICH-G2 [183],

for cross-site runs on resources across the NGS (UK) [240], TeraGrid (US) [241] and LONI (US) [188] (see Chapter 3). The use of HARC (Highly Available Robust Co-scheduler) for advance reservations plays a crucial role in surgical treatment planning, along with SPRUCE (Special PRiority and Urgent Computing Environment) for urgent computing purposes. These various aspects have been combined together within the Application Hosting Environment (AHE) [242–244] to support automated launching of applications onto various computational resources.

In this work, we address the use of computational grids and in particular federations of these grids, such as the TeraGrid and the UK National Grid Service (NGS) [240] to conduct patient-specific simulation of the intracranial vasculature, conducted as part of the GENIUS (Grid Enabled Neurosurgical Imaging Using Simulation) project¹ [245]. This project is being conducted in collaboration with consultant radiologists at the National Hospital for Neurology and Neurosurgery (NHNN) in London, U.K.

GENIUS requires access to computational resources in a fashion that goes beyond the typical batch job submission scenario which is the standard access mechanism offered by the majority of HPC service providers. The existing policies at these sites are not suitable for medical applications as they stand. To support patient-specific medical simulations, where life and death decisions might be made using a grid-computing platform, new modes of computation need to be introduced. Similar to booking and prioritising laboratory pathology testing, simulations can be given urgent priorities, or they can be booked in advance [246, 247]. Furthermore, the ability to perform this type of computing allows a user to be physically available when the simulation takes place, which is essential for interactivity purposes.

Since the scientist or the clinician is at an arbitrary geographical distance from where the simulation and visualisation are carried out, computational steering and high performance parallel capabilities have been added to HemeLB, to allow for the remote interactive exploration of cerebral blood flow (see Chapter 4).

This chapter begins by outlining our infrastructure in Sec. 5.2, detailing its various features which include advance reservations, emergency computing, the Application Hosting Environment, cross-site runs and distributed computing, computational steering and dedicated lightpath networks. HemeLB is presented in Chapter 3 while the visualisation approach is described in detail in Chapter 4. Then, the use of these grid technologies in a clinical context, the acquisition of patient-specific neurovasculatures, the anonymisation of patient datasets, and the clinical workflow are discussed in Sections 5.3.1, 5.3.2 and 5.3.3 respectively.

¹GENIUS [245] is a joint US/UK high-end computing project, funded by the UK EPSRC & US NSF.

5.2 Grid infrastructure

In working with consultant neuroradiologists, there are various clinical work-flow aspects which need to be adhered to. Probably of most importance is the timeliness, where results from medical imaging to blood flow simulation and visualisation must be obtained within a few weeks to 10-30 minutes in extreme emergency situations (as stated by clinicians). Other factors include the transparency of the software interface, since the details concerning the underlying computational infrastructure are not of concern to clinicians. Some of these issues have been addressed in GENIUS through extensive collaboration and discussions with clinicians, as well as observation of their work-flow practices.

GENIUS requires that clinicians can access multiple machines interactively, to steer and visualise simulations in a time frame that is clinically relevant. To achieve the required turn around times such simulations cannot be run in normal batch queues; they need to be given a higher priority, and require some form of on-demand computing to succeed.

These requirements lead to a demand on resource providers to implement policies and tools that allow computational access to be gained as and when required, so that such methodologies can be incorporated into a clinician's day to day activities, rather than just providing such facilities on an *ad hoc* basis [246].

The GENIUS project also has advanced networking needs, typically requiring resources to be connected with dedicated lightpath links in order to perform inter-machine simulations and facilitate real-time steering and visualisation. In addition to these hardware requirements, suitable middleware tools are needed to hide the components of the grid from researchers and clinicians. We discuss the infrastructure requirements of GENIUS in greater detail in the following sections.

5.2.1 Advance reservations

To interact with a blood flow simulation and explore its results in real time, a scientist or a clinician must be physically present during the corresponding run; therefore, if the computational resource exploited for the simulation are shared by other users he/she should be able to reserve some minutes or hours at a time in which he/she can monitor and control the simulation. Implementing an advance reservation system across a single grid entails handling some difficulties: each grid has its own policies and systems for making advance reservations, if it has any at all. Additionally, the high performance network provision between grids may also be limited.

A few systems exist to allow users to co-reserve time on grid resources. GUR (Grid Universal Remote) [248, 249] is one such system, developed at San Diego Supercomputer Center (SDSC) [250].

HARC, which was developed by Dr. Jon MacLaren, is a widely deployed open-source system that allows users to reserve multiple distributed resources in a single step [234]. HARC is employed within GENIUS on a regular basis to make single and multiple machine reservations to conduct interactive blood flow simulations and visualisations.

5.2.2 Emergency computing

As well as the need to reserve computational resources in advance, GENIUS makes use of so-called *urgent computing* middleware in order to gain immediate access to resources in emergency situations. Such middleware allows a high priority job, in this case a brain blood flow simulation, to pre-empt the applications running on a machine. This model applies to slightly different situations than the advance reservation case; the latter would be of most use when a clinician knows in advance that a simulation needs to be performed at a specific time. The former model is most useful when a medical simulation needs to be performed urgently, but the need for the simulation is not known in advance.

SPRUCE [247] is an urgent computing solution that has been developed to address the growing number of problem domains where critical decisions must be made quickly with the aid of large-scale computation. HemeLB has been used with SPRUCE on the Lonestar cluster at the Texas Advanced Computing Center (TACC), and demonstrated live on the show floor at Super Computing 2007, where real-time visualisation and steering were employed to control HemeLB within an urgent computing session.

5.2.3 The Application Hosting Environment

Hospital clinicians and technicians, have no experience using high performance compute resources and computational grids. To prevent them from having to learn and deal with a large stack of different middleware tools in order to execute the required data processing and simulation workflow, a clinical interface based on the Application Hosting Environment [242–244] has been developed.

The Application Hosting Environment (AHE) is a lightweight mechanism for representing scientific applications, and allowing users to interact with those applications using simple client tools. AHE enables the launching of hosted applications on a variety of different computational resources, from national and international grids of supercomputers, through institutional and departmental clusters, to single processor desktop machines. It does so transparently, meaning that the end user is presented with a single interface and access mechanism to launch applications on all of these resources. AHE also provides mechanisms for file transfer and job management, allowing the user to move input and output data between their desktop and target

computational resource, and to monitor or terminate applications as they run. For a fuller discussion see [243, 244].

Within GENIUS, AHE is used to host the HemeLB code at target sites on TeraGrid, LONI and UK NGS, and to launch simulations using data derived from a patient. We have simplified the AHE's GUI to better support the needs of clinical users.

5.2.4 Distributed computing

HemeLB can employ a small number of processor cores from a number of machines in order to run a simulation, rather than requiring a large number of cores from a single machine, which substantially increases the potential number of target resources available, and therefore the number of clinical simulations that can be conducted at any one time.

HemeLB has been combined with MPIg, the latest release of the successor to MPICH-G2 [183], a version of MPI that allows simulations to be distributed across multiple resources, using the Globus middleware for inter-machine communication. MPIg hides latency much more effectively than its predecessor. The communication costs of geographically distributed domain decomposition being overlapped with computation become almost negligible, meaning that cross-site runs with HemeLB on large models run very efficiently [184]. A requirement of running MPIg models is that cross-site reservations of compute time can be made on all of the resources on which the problem is being distributed; we use HARC for this.

5.2.5 Computational steering

Computational steering [221] allows a user to make more efficient use of computational resources by providing a means by which he/she can remotely interact in real time with a simulation. By monitoring the progress of simulations, aided by on-line visualisation, the computational scientist avoids losing cycles due to redundant computation or even doing the wrong calculation. By tuning the value of steerable parameters, the scientist quickly learns how the simulation responds to perturbations and can use this insight to design subsequent computational experiments.

In the case of the GENIUS project, computational steering is currently being used to control simulation and *in situ* visualisation parameters through asynchronous socket-based communications between the server and the client (see Chapter 4).

The scientist employs an in-house graphical user interface (GUI) running on the 'steering client' to view and set parameters or stop the running simulation; a snapshot of the GUI interface which controls HemeLB's execution is given in Fig. 5.1. It enables control of view parameters *e.g.* viewpoint position and zoom-factor, and

verify performance and flow field characteristics like the average inlet velocity and the maximum wall shear stress (see Chapter 6 for further details).

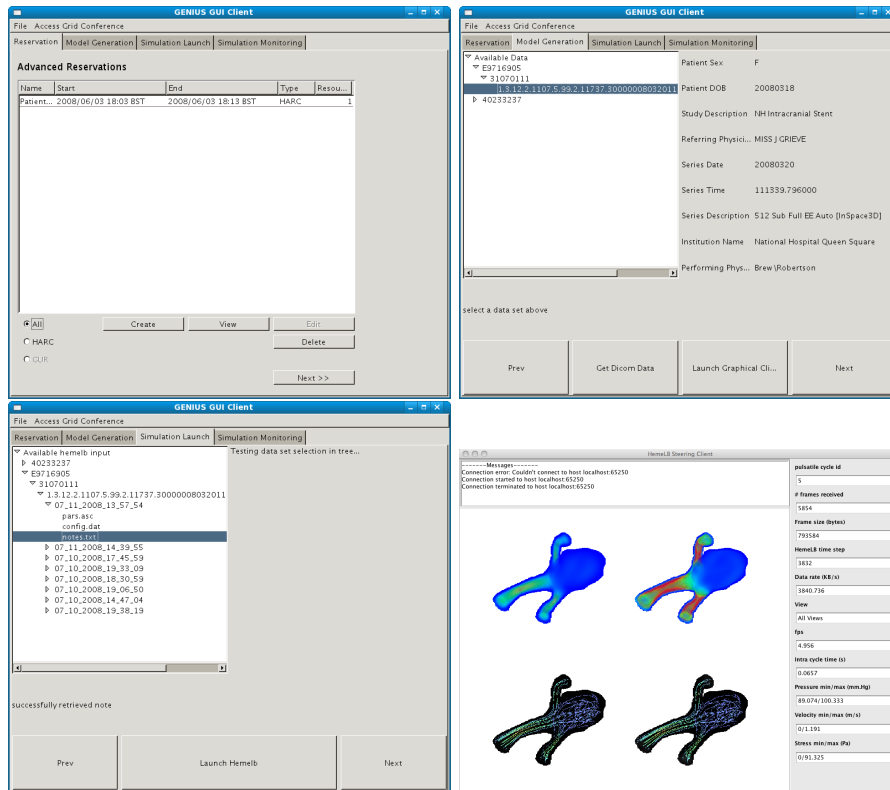


Figure 5.1: GENIUS GUI screenshots. The GENIUS GUI co-reserves a certain number of hours with HARC (top left), generates the computational model (top right), launches the corresponding simulation (bottom left) and interactively monitors and steers HemeLB (bottom right).

5.2.6 Dedicated networking with lightpaths

Computational grids are usually connected using shared TCP/IP networks. Within GENIUS, we seek to use low-latency, high-bandwidth optical networks wherever possible. These lightpaths provide several features that are not achievable using regular, best-effort networks, but which are needed for high performance grid computing. These are not necessary within networks such as the TeraGrid or LONI, since they are already interconnected with dedicated high-speed networks. However, to shift data between these remote resources to local facilities, such as the angiography suite at the NHNN, dedicated lightpaths are required. They are also ideal to support distributed and interactive simulation and visualisation.

GENIUS makes use of a purpose-built dedicated 1 Gb/s lightpath network which spans the UK and US, designed to culminate on a desktop workstation within the angiography suite at the National Hospital for Neurology and Neurosurgery in Queen’s Square, London (Fig. 5.2). From within the suite control room, clinicians can inter-

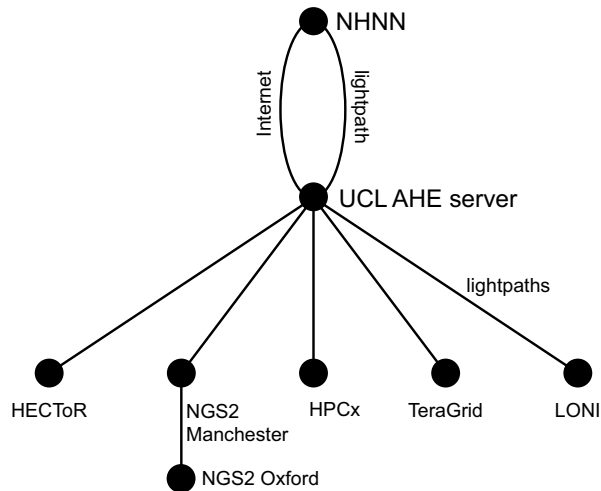


Figure 5.2: The GENIUS lightpath network. Dedicated gigabit links are used to connect computational infrastructure such as the TeraGrid and LONI directly to the NHNN angiography suite. In the case of the NGS2 Manchester and Oxford nodes, a dedicated link is used to facilitate cross-site MPIg runs, with the NGS2 Manchester \rightarrow UCL link used for steering, visualisation and medical data transfer.

actively run, steer and visualise subject-specific neurovascular blood flow on remote resources using the contention-free lightpath network.

The topology-aware data communications implemented in HemeLB take advantage of the asynchronous communication capability provided by MPIg which hides latency much more effectively than earlier MPI versions suitable for cross-site runs; as a consequence, the corresponding simulations run very efficiently (see Fig. 5.3 and Chapter 3 for further details).

Visualisation of fluid flow fields has been traditionally addressed as a post-processing step. The drawback here is that large amounts of data must be transferred between different platforms in different locations, requiring large storage capacities, high bandwidths and big local I/O. The rendering method adopted is a sort-last ray tracing technique in which each processor-subdomain is rendered independently. See Chapter 3 and Chapter 4 for all the technical details about the fluid solver and the *in situ* visualisation approach of HemeLB respectively.

5.3 Medical simulation in the operating theatre

In bridging the gap between computational science research and its use in the operating theatre, various clinical and scientific issues need to be addressed which are discussed in the following sections.

5.3.1 Acquiring patient-specific neurovasculatures

During angiography, the clinician uses a contrast enhancing fluid, injected into the blood stream, to examine the blood flowing through the vasculature. A recent advance in medical imaging is three-dimensional rotational angiography (see Chapter 1). The scanner, in this case a Siemens Artis Zee [251], has recently been installed in the new angiography suite at the NHNN in Queens Square, London. Using standard image segmentation techniques [252] the three-dimensional vasculature can be extracted from these high-contrast projections and used as the input configuration to HemeLB. We have developed an efficient graphical-editing tool to quickly reconstruct and manipulate the medical datasets needed by HemeLB (see Chapter 6 for details).

5.3.2 Data anonymisation

In using high performance computing resources on a federated intercontinental grid for medical simulation, data will not only cross administrative boundaries from within the hospital network to the (untrusted) outside world, but also international borders. Thus, data needs to be suitable anonymized. Thus, an important aspect of this work involves development of methods by which data can be anonymized within the hospital network, before transfer over high-speed network links to remote intercontinental computing resources.

5.3.3 The clinical work-flow

The software environment developed within GENIUS aims at hiding unnecessary grid-computing details from clinicians, whilst bringing to the forefront the details and processes which clinicians need to be aware of. These include (i) data acquisition, (ii) the process of image segmentation to obtain a 3D model of the neurovascular structure, (iii) the specification of boundaries, and their pressure or velocity conditions, and (iv) interactions with the real-time rendered simulation. The clinical grid computing interface relies on the GENIUS desktop client, designed to easily handle and automate all those steps (see Fig. 5.1).

Clinicians are not concerned with where simulations are running, nor the details of advance reservations; thus features such as advance reservations and emergency computing capabilities, job launching and resource selection are done behind the scenes. Fig. 5.4 shows the clinical work-flow from the clinician's perspective. This environment is particularly important given the time scales involved in the clinical decision making process. From the acquisition of a 3DRA dataset to the corresponding treatment, a time scale of 10 to 30 minutes occurs often in emergency situations, and the workflow has to adapt to this. The patient-specific medical-simulation process within GENIUS is as follows:

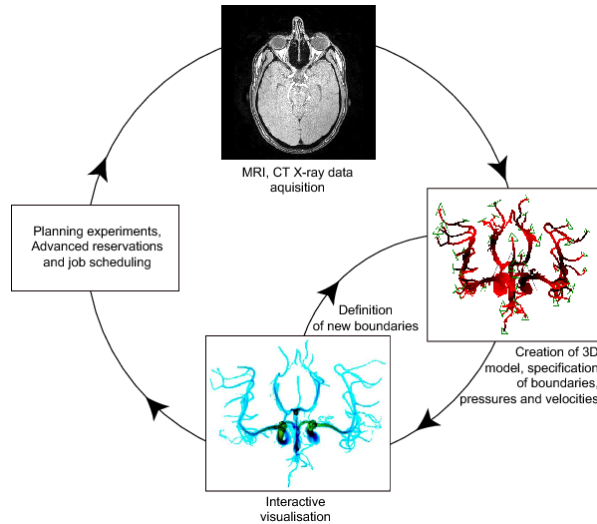


Figure 5.3: Our clinical workflow. The patient-specific dataset is acquired through a medical scanner. Then, the dataset is segmented and manipulated to impose blood flow conditions at the boundaries of the selected region of interest. The corresponding data are migrated to a supercomputer where the blood behaviour is simulated and rendered during a HARC pre-reserved slot. The simulation is interactively monitored and controlled by means of the steering capabilities implemented in HemeLB and our graphical user interface (see 5.3.3 for further details).

1. A dataset derived from a patient’s scan is anonymised locally in the hospital and uploaded from the medical imaging workstation to a DICOM server (co-located with the ‘UCL AHE server’), which is dual-homed on both the UCL production and lightpath network. The two networks are physically separate in order to adhere to NHNN network security policies.
2. The client software connects to the DICOM server, giving the clinician the ability to browse and download a patient dataset.
3. The data is segmented into the three-dimensional neurovasculature and boundary conditions are applied by a clinician or an operator. These can be patient-specific values measured by a clinician; alternatively, typical values may be used.
4. The data files which describe this system and other patient-specific settings are staged using AHE to a remote computational resource.
5. The job can be executed within a pre-reserved slot which has been booked using HARC, as a cross-site or single-site run.
6. As the job commences running, the client connects to the back end node hosting rank 0, which is used for steering and visualisation purposes.

Notably, it would be impossible to compress the above steps into the required clinical

time frame without the availability of an urgent computing system, or pre-reserved slots on remote resources.

5.4 Discussion

In this chapter, we have discussed middleware which brings together a number of capabilities to enable the use of computational grids for patient-specific cerebral blood flow simulation. HemeLB is at the core of the GENIUS project, and its efficiency and scalability entails simulating complex pulsatile blood flows rapidly enough to fit into the urgent timeframe of a real clinical scenario since the velocity, stress and pressure fields can be visualised in real-time, without the need for post-processing (see Chapter 4 and 6)². The long-term objective is to determine the direct clinical utility not only of cerebral haemodynamics, but also real-time patient-specific medical simulation methods in general, where clinical studies will be required.

In achieving the goal of using simulation within a clinical environment, various computing modalities need to be employed: advance reservations, urgent capabilities, distributed and grid computing, the Application Hosting Environment, real-time visualisation, steering and fast lightpath networks. These modalities are necessary prerequisites to achieve the crucial medical factor of timeliness of results. We hope that our effort will contribute to the realisation of a persistent and robust system through which grid computing will be adopted for routine use within a clinical environment.

²The flow field can still be output on files at regular time intervals for advanced post-processing. Several images are usually written to files for visual feedback.

Chapter 6

Fluid flow simulation in simple and patient-specific systems

*Imagination is the beginning of creation. You imagine what you desire, you will
what you imagine and at last you create what you will*

George Bernard Shaw

In this chapter, we present various simulation results. First, we test the accuracy of different boundary condition methods (BCMs) by simulating fluid flow dynamics within two-dimensional channels, three-dimensional square ducts and cylinders. The two-dimensional results witness that the LBM is of a second-order accuracy in time while the error dependent on the lattice resolution is much smaller than that connected to the Mach number if the latter is not very low. Therefore, the accuracy is very high for simulations with low local velocities (in the order of 0.01 in lattice units). In three dimensions and with curved boundaries, the application of BCs available in the literature as well as those proposed in Chapter 3 prevents achieving the intrinsic accuracy of the LBM mentioned above.

In the second part of this chapter, computational investigations of patient-specific neurovascular pathologies performed by means of the grid computing capabilities described in Chapter 5 are presented in detail. Specifically, time-varying blood flow behaviour were simulated with the fluid solver HemeLB (Chapter 3) and visualised through the *in situ* visualisation techniques described in Chapter 4.

6.1 Simulations of fluid flow in simple geometries

In this section, simulation results of fluid flowing through two-dimensional channels and three-dimensional square ducts and cylinders are presented in order to assess the accuracy of the LBGK model (see Chapter 3) as a function of spatial and temporal resolutions, Mach number (Ma) and the use of different BCMs.

First, two-dimensional results of a steady fluid flow show that it is possible to recover the analytic velocity description within machine accuracy when an incompressible LBGK model and a BCM, which is consistent with the mathematical formulation of that model, are employed. The other two-dimensional simulation results confirm that the LBM is of a second-order accuracy in time and that the error is also proportional to Ma^2 .

Then, three-dimensional results obtained with a LBGK model are presented to test the accuracy of different boundary condition approaches; specifically, the BCM used in HemeLB and described in Chapter 3 is simple, efficient and its accuracy is superior other local BCMs [98]; its variant (Chapter 3), instead, is non-local but more accurate and still simple to implement.

6.1.1 Two-dimensional results

Stationary and time-varying Newtonian fluids flowing within a two-dimensional channel are simulated through the LBGK models coined D2Q9 [101] and its incompressible counterpart D2Q9i [105].

We employed the BCM proposed by Zou and He [253]. It can be only applied to planar boundaries perpendicular to the Cartesian axes (see discussion below). For a pressure-controlled boundary with pressure $\bar{p} = \bar{\rho}c_s^2$, the inward pointing distribution functions and macroscopic velocity $\mathbf{u}(t + \Delta t)$ are computed by means of the following system of equations:

$$\begin{cases} \bar{p} = \sum_i f_i(\mathbf{x}, t + \Delta t), \\ \bar{\rho}\mathbf{u}(t + \Delta t) = \sum_i \mathbf{e}_i f_i(\mathbf{x}, t + \Delta t), \\ f_{i^+}(\mathbf{x}, t + \Delta t) - f_{i^+}^{(eq)}(\mathbf{x}, \mathbf{u}(t + \Delta t)) = f_{\bar{i}^+}(\mathbf{x}, t + \Delta t) - f_{\bar{i}^+}^{(eq)}(\mathbf{x}, \mathbf{u}(t + \Delta t)), \end{cases} \quad (6.1)$$

where i^+ is the inward pointing distribution function (see Chapter 2 for a discussion about BCMs and definitions) perpendicular to the pressure boundary and $\mathbf{e}_{\bar{i}^+} = -\mathbf{e}_{i^+}$. The last equation is the bounce-back principle while the others are formulae of the LBM¹. The total equations are six and if the unknown values are more than the equations the system can only be solved if one employs some extra rules. Therefore,

¹This approach is referred to as ‘‘consistent’’ BCM because of the employment of the LB equations themselves plus the bounce-back rule.

the original formulation of this BCM cannot be applied to any boundary configuration. For a pressure boundary perpendicular to a Cartesian axis we assume that the velocity components parallel to the boundary are zero. In this case, for the D2Q9, D2Q9i and D3Q15 LBGK models the system in question can be solved and is explicit. A velocity boundary condition is approached in the same manner. To this end, we are able to approach any boundary lattice site of a two-dimensional rectangular channel whose pressure and zero-velocity boundaries are aligned with the Cartesian axes except the four corner lattice sites (see Fig 6.1). Here, the pressure and the velocity are both controlled but the inward pointing distribution functions are five and the equations are four. To circumvent the problem, Zou and He [253] proposed to assume that the inward pointing distribution functions depicted at the left and topmost corner of Fig 6.1) are equal. Below, the accuracy of the LBGK models D2Q9 [101] and D2Q9i [105] is studied as a function of Ma , spatial and temporal resolutions.

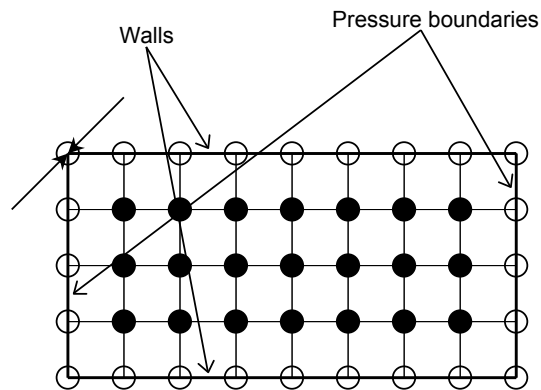


Figure 6.1: Two-dimensional illustration of a lattice within a channel of size (L_x, L_y) (L_y is the height) whose fluid flow is confined by the horizontal no-slip walls and is driven by the pressure-controlled vertical boundaries. The boundary sites are denoted by empty circles. Note that the total fluid lattice sites are $(L_x + 1) \times (L_y + 1)$.

Steady fluid flow

The fluid within a rectangular channel of dimensions (L_x, L_y) with no-slip wall condition is driven by a pressure gradient $-\Delta P$ which is constant in time along the axis X parallel to the centerline of the channel. The fluid flow in the laminar regime is described by the Poiseuille formulae:

$$\frac{\partial p}{\partial x} = -\Delta P, \quad \frac{\partial p}{\partial y} = 0, \quad (6.2)$$

$$u_y = u_0 \left(1 - \frac{4y^2}{L_y^2}\right), \quad u_x = 0, \quad (6.3)$$

$$p(x) = p_i - \frac{2x + L_x}{4} \Delta P, \quad (6.4)$$

where u_x and u_y are the velocities along axes X and Y respectively, $p(x)$ is the pressure along the vertical line located at position x , $p_i = c_s^2 + \Delta P \frac{L_x}{2}$ that at the inlet, u_0 is the centerline velocity (peak velocity) and

$$\Delta P = \frac{8\nu u_0}{L_y^2}, \quad (6.5)$$

The initial condition for each lattice site is given by the equilibrium distributions calculated with a unitary density and zero velocity and the steady-state is considered to be attained if

$$\frac{\sum_i \sum_j \sqrt{(u_x(i, j, t) - u_x(i, j, t-1))^2 + (u_y(i, j, t) - u_y(i, j, t-1))^2}}{\sum_i \sum_j \sqrt{u_x(i, j, t)^2 + u_y(i, j, t)^2}} \leq Tol \quad (6.6)$$

where $Tol = 10^{-10}$ and (i, j) refers to a lattice site location. The error in the velocity field with respect to the analytical solution $(u'_x, u'_y)(i, j)$ is

$$\xi_u = \frac{\sum_i \sum_j \sqrt{((u'_x - u_x)(i, j))^2 + ((u'_y - u_y)(i, j))^2}}{\sum_i \sum_j \sqrt{(u'_x(i, j))^2 + (u'_y(i, j))^2}}, \quad (6.7)$$

while the error in the density field is

$$\xi_\rho = \frac{\sum_i \sum_j |(\rho' - \rho)(i, j)|}{\sum_i \sum_j |\rho'(i, j) - 1|}, \quad (6.8)$$

where $\rho'(i, j)$ is the analytical solution. Accuracy results for $Re = 1.28$ ($Re = \frac{u_0 L_y}{\nu}$), $\nu = 0.05$ and $u_0 = 0.064\Delta x$ (the lattice resolution is $\Delta x = 1/L_y$, $L_x = 2L_y$) are given in Table 6.1 while the same results but at Re and u_0 ten times higher are provided in Table 6.2.

By combining the LBGK model D2Q9i and the BCM presented by Zou and He [253] we achieved a formidable accuracy which can also be increased by decreasing Tol (not shown). The number of time steps needed to achieve convergence is clearly proportional to $1/\Delta x^2$ when Ma is decreased while it scales as Δx if Ma is kept constant. The errors obtained with the D2Q9 and $Re = 12.8$ (Table 6.2) are an order of magnitude higher than those attained at $Re = 1.28$ (Table 6.1) for a certain spatial resolution. This suggests that the error is dominated by Ma –not by Δx . To clarify this ambiguity, we conducted other simulations at a constant Re and Ma by using the D2Q9 model. The corresponding results are shown in Table 6.3.

		$\Delta x = 1/4$	$\Delta x = 1/8$	$\Delta x = 1/16$	$\Delta x = 1/32$	order
D2Q9	ξ_ρ	9.42e-6	3.37e-7	1.49e-8	8.46e-10	4.48
	ξ_u	2.72e-3	6.42e-4	1.56e-4	3.85e-5	2.05
	ts	643	2469	9237	34151	
D2Q9i	ξ_ρ	2.05e-11	8.48e-12	3.64e-12	6.97e-12	MA
	ξ_u	1.03e-9	4.34e-9	1.74e-8	6.93e-9	MA
	ts	643	2469	9237	34153	

Table 6.1: Errors, orders of accuracy and number of time steps (ts) for the two-dimensional Poiseuille flow simulated with the LBGK models coined D2Q9 and D2Q9i at $Re = 1.28$ with $u_0 = 0.064\Delta x$ and $\nu = 0.05$. “MA” means that accuracy similar to the machine precision is achieved. See text for further details.

		$\Delta x = 1/4$	$\Delta x = 1/8$	$\Delta x = 1/16$	$\Delta x = 1/32$	order
D2Q9	ξ_ρ	2.28e-3	1.89e-4	1.25e-5	7.94e-7	3.84
	ξ_u	4.56e-2	1.27e-2	3.26e-3	8.21e-4	1.93
	ts	656	2443	9283	34573	
D2Q9i	ξ_ρ	3.58e-11	1.34e-11	6.51e-12	4.44e-12	MA
	ξ_u	6.73e-10	4.83e-9	1.56e-8	5.64e-8	MA
	ts	656	2455	9291	34581	

Table 6.2: Errors, orders of accuracy and number of time steps (ts) for the two-dimensional Poiseuille flow simulated with the LBGK models coined D2Q9 and D2Q9i at $Re = 12.8$ with $u_0 = 0.64\Delta x$ and $\nu = 0.05$. “MA” means that accuracy similar to the machine precision is achieved. See text for further details.

		$\Delta x = 1/4$	$\Delta x = 1/8$	$\Delta x = 1/16$	$\Delta x = 1/32$
$Re = 1.28$ $u_0 = 0.016$	ξ_ρ	9.42e-6	3.00e-6	2.97e-6	3.16e-6
	ξ_u	2.72e-3	2.57e-3	2.50e-3	2.46e-3
	ts	643	1285	2501	4833
$Re = 12.8$ $u_0 = 0.16$	ξ_ρ	2.28e-3	2.54e-3	2.79e-3	2.93e-3
	ξ_u	4.56e-2	4.67e-2	4.69e-2	4.70e-2
	ts	656	1273	2544	4953

Table 6.3: Errors and number of time steps (ts) for the two-dimensional Poiseuille flow simulated with the LBGK model coined D2Q9 with $\nu = 0.2\Delta x$. See text for further details.

It is evident that the error in the computation of the velocity field is controlled by Ma –not by Δx ; the behaviour of the error in the density field is similar but when $Re = 1.28$ we have $\xi_\rho(\Delta x = 1/4) \approx 3\xi_\rho(\Delta x = 1/8)$. This means that ξ_ρ is governed by Δx in this case only. This is confirmed by the fact that ξ_ρ continued to grow by maintaining $Re = 1.28$ and $u_0 = 0.016$ whilst decreasing Δx to $1/2$: $\xi_\rho(\Delta x = 1/2) = 2.50 \times 10^{-5}$ versus $\xi_\rho(\Delta x = 1/4) = 9.42 \times 10^{-6}$ (the error in the velocity field is still constant: $\xi_u(\Delta x = 1/2) = 0.00293 \approx \xi_u(\Delta x = 1/4) = 0.00273$).

In conclusion, the results provided above show that the LBM is intrinsically very accurate for the simulation of steady flow within a rectilinear channel in two dimensions. The error is dominated by Ma^2 and does not significantly depend on the spatial resolution.

Unsteady fluid flow

Now, a similar study is presented to analyse the error of the LBM as a function of temporal resolution. While steady flows are characterised by Re , the time-varying fluid flow description depends on Re as well as the Womersley parameter $\alpha = \frac{L_y}{2} \sqrt{2\pi/(T\nu)}$ [21], where T is the time period of the time-dependent boundary pressure (or body force) which drives the fluid.

The current investigation analyses the accuracy in time of the D2Q9 and D2Q9i models by maintaining Re and α constant. The fluid flow in the laminar regime under a pressure gradient $-\Delta P(t) = -\Delta P_0 \cos(\omega t + \pi/2)$ is described by the following formulae:

$$u_y(t) = \frac{4\Delta P_0}{\rho\pi} \sum_{m=0}^{\infty} \frac{(-1)^m}{2m+1} \cos \frac{(2m+1)\pi y}{L_y} \left\{ \frac{\nu \frac{\pi^2}{4} [m] \cos(\tilde{\omega}) + \omega \sin(\tilde{\omega})}{\omega^2 + \nu^2 (\frac{\pi^4}{16}) [m]^2} \right\} \quad (6.9)$$

$$u_x(t) = 0, \quad (6.10)$$

$$p(x, t) = p_i(t) - \frac{2x + L_x}{4} \Delta P(t), \quad (6.11)$$

where $\omega = 2\pi/T$, $\tilde{\omega} = \omega t + \frac{\pi}{2}$, $u_x(t)$ and $u_y(t)$ are the velocities along axes X and Y respectively, $p(x, t)$ is the pressure at position x and time t , $p_i(t) = c_s^2 + \Delta P(t) \frac{L_x}{2}$ and $[m]$ is an abbreviation for

$$\left[\frac{(2m+1)^2}{(L_y/2)^2} \right]. \quad (6.12)$$

$u_y(t)$ is the velocity profile provided by Fan and Chao [254] for a two-dimensional channel. The simulation is assumed to be converged if

$$\frac{1}{T} \sum_t \frac{\sum_i \sum_j \sqrt{(u_x(i, j, t) - u_x(i, j, t-T))^2 + (u_y(i, j, t) - u_y(i, j, t-T))^2}}{\sum_i \sum_j \sqrt{u_x(i, j, t)^2 + u_y(i, j, t)^2}} \leq Tol, \quad (6.13)$$

		$T = 1e + 3$	$T = 2e + 3$	$T = 4e + 3$	$T = 8e + 3$	order
Re = 0.664	ξ_u^T	4.06e-3	9.69e-3	2.68e-3	9.28e-4	1.82
	cycles	8	8	8	8	
Re = 6.64	ξ_u^T	4.90e-3	1.11e-2	2.87e-3	9.18e-4	1.89
	cycles	8	8	8	8	
Re = 66.4	ξ_u^T	unstable	2.27e-1	6.24e-2	1.56e-2	1.93
	cycles	8	8	8	8	

Table 6.4: Errors and number of time cycles for the two-dimensional time-varying fluid flowing through a rectilinear channel simulated with the LBGK model coined D2Q9. $\nu = 400\Delta t$ while $u_{max} = 8.3\Delta t$, $u_{max} = 83\Delta t$ and $u_{max} = 830\Delta t$ for Re equal to 0.664, 6.64 and 66.4 respectively. See text for further details.

where $Tol = 10^{-10}$. The simulation error in the velocity field with respect to the analytic solution (u'_x, u'_y) were calculated by means of the following formula:

$$\xi_u^T = \frac{1}{T} \sum_t \frac{\sum_i \sum_j \sqrt{((u'_x - u_x)(i, j, t))^2 + ((u'_y - u_y)(i, j, t))^2}}{\sum_i \sum_j \sqrt{(u'_x(i, j, t))^2 + (u'_y(i, j, t))^2}}. \quad (6.14)$$

The flow fields during the current and previous time cycles are advanced in time and compared through the method presented in Chapter 3.

The simulation error in the velocity field as a function of temporal resolution $\Delta t = 1/T$ is given in Table 6.4 for various Reynolds numbers by using the D2Q9 model, $\Delta x = 1/32$ and $\alpha = 2.0053$.

The D2Q9 model is second-order accurate in time. The number of cycles needed to converge does not depend on Δt or Ma. For a certain temporal resolution the simulation error for Re = 0.664 is very similar to that attained with Re = 6.64, while the error corresponding to Re = 66.4 is an order of magnitude higher. This means that the error is proportional to Δt^2 for low Mach numbers and not very low temporal resolutions, while it depends on Ma^2 for high velocities. Specifically, from the caption of the table we can see that the accuracy becomes higher if the maximum velocity is lower than a critical value of about 0.1 in lattice units. The D2Q9i model is less sensitive to Ma. In fact, for Re = 66.4 ξ_u^T is equal to 3.49×10^{-2} , 7.41×10^{-3} and 9.20×10^{-4} for $\Delta t = 1/2000$, $\Delta t = 1/4000$ and $\Delta t = 1/8000$ respectively, which means that the critical velocity in lattice units is ≈ 0.2 .

6.1.2 Three-dimensional results

We implemented and studied various BCs including the approaches presented in Chapter 3 and a few ones available in the literature. We tested all these techniques in conjunction with the three-dimensional LB model called D3Q15 (see Chapter 3).

Specifically, we implemented the BCM presented by Zou and He [253] and that of Yang [171] to handle pressure-controlled condition at planar axis-aligned boundaries, and the BCM presented by Guo *et al.* [6] for generic pressure and velocity BCs; the no-slip wall condition was also applied using the bounce-back rule. We simulated fluid flowing through axis-aligned square ducts and cylinders whose centerline is parallel to a Cartesian axis or inclined. We provide accuracy results after reviewing the aforementioned BCMs.

Zou and He’s boundary condition method

The BCM proposed in [253] was extensively described in Sec. 6.1.1 where we showed that, in conjunction with the D2Q9 and D2Q9i models, the Poiseuille fluid flow within a axis-aligned channel can be recovered within a high accuracy for non-high Mach numbers. Here, we emphasise that, in three dimensions and with the D3Q15 model, this method cannot be applied to the lattice sites of a complex irregular boundary or the edges of a square duct because the number of inward pointing distribution functions (see Chapter 2 for some BC-related concepts and terminology) is larger than the available equations (see [253] for further details). We do not assume extra rules; hence, we only apply the method to the lattice sites positioned at the pressure boundaries and which are not in contact with the no-slip walls. For these corner lattice sites the inward pointing distribution functions are set as follows:

$$f_i(\mathbf{x}, t + \Delta t) = f_i^{(eq)}(\bar{p}, \mathbf{0}) + f_i^{(neq)}(\mathbf{x}, t), \quad (6.15)$$

where \bar{p} is the pressure boundary at position $\bar{\mathbf{x}}$ and time $t + \Delta t$. As discussed in Chapter 3, the error related to the fact that, in general, \mathbf{x} does not coincide with the wall is $O(\Delta x \text{Ma})$ while the error due to the contribution of the non-equilibrium part is $O(\Delta t^2)$. The BCM presented by Zou and He was employed in conjunction with the bounce-back method for the simulation of fluid flow in a square duct where the wall was at half way through the lattice vectors, that is, $q_u = 1/2$ in Fig. 6.2. In this case, however, the aforementioned BCM does not spoil the second-order accuracy of the bounce-back rule (see [62]), as shown in the simulation results (below). The application of the bounce-back method to wall lattice sites controlled by the pressure gave unstable results: at every time step, the velocity flow field substantially oscillated in magnitude. This phenomenon is known as “checkerboard effect” [255]; notwithstanding Kandhai *et al.* [255] noted that the D3Q19 model is not significantly influenced by this numerical issue, we found out that, in our case, the situation did not improve by replacing the D3Q15 model with the D3Q19 one.

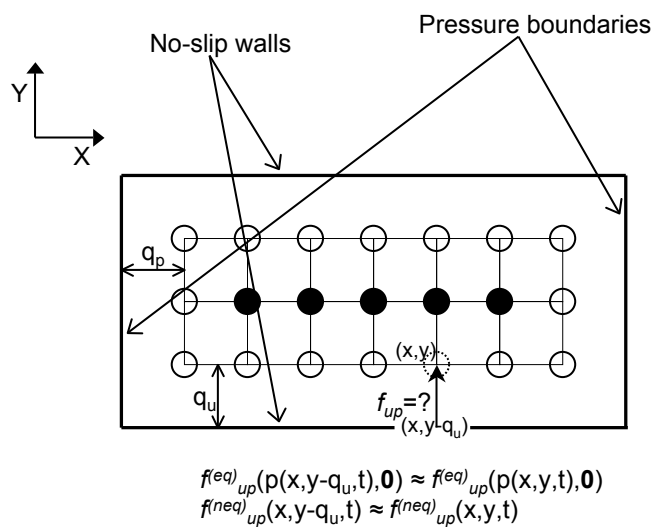


Figure 6.2: Two-dimensional illustration of a lattice within a channel of size (L_x, L_y) (L_y is the height) whose fluid flow is confined by the horizontal no-slip walls and is driven by the pressure-controlled vertical boundaries. The lattice sites close to the boundaries are denoted by empty circles. Note that the total fluid lattice sites are $(L_x - 1) \times (L_y - 1)$ if $q_u = q_p = 1$. At the bottom, we show the approximations employed by Guo *et al.* [6] to approach pressure-controlled boundaries (see text for further details).

Guo *et al.*'s boundary condition method

We also explored the BCs proposed by Guo *et al.* [6]. We refer to Fig. 6.2 for a schematic illustration in two dimensions. The the post-collision inward pointing distribution function at coordinates $(x, y - 1)$ is

$$f_{up}^+(x, y - 1, t) \equiv f_{up}^{(eq)}(x, y - 1, t) + \left(1 - \frac{1}{\tau}\right) f_{up}^{(neq)}(x, y - 1, t) \quad (6.16)$$

obtained by decomposing the distribution function into its equilibrium and non-equilibrium parts, but $p(x, y - 1, t)$, $\mathbf{u}(x, y - 1, t)$ and $f_{up}^{(neq)}(x, y - 1, t)$ are not defined. Guo *et al.* [6] proposed to use the following approximations:

$$f_{up}^{(neq)}(x, y - 1, t) = f_{up}(x, y, t) - f_{up}^{(eq)}(x, y, t) + O(\Delta x^2), \quad (6.17)$$

$$\mathbf{u}(x, y - 1, t) = \mathbf{u}(x, y, t) + O(\Delta x \text{Ma}) \text{ (pressure boundary)}, \quad (6.18)$$

$$p(x, y - 1, t) = p(x, y, t) + O(\Delta x \text{Ma}^2) \text{ (no-slip wall)}. \quad (6.19)$$

We note that a boundary with pressure \bar{p} , in general, is not located at position $(x, y - 1)$; therefore $p(x, y - 1, t) = \bar{p} + O(\Delta x \text{Ma}^2)$. A similar discussion holds for a generic velocity boundary. To void introducing the last approximation whilst simulating fluid flow within a square duct whose boundaries are perpendicular to the Cartesian axes, we impose $q_u = q_p = 1$ (see Fig. 6.2) so as to have $p(x, y - 1, t) \equiv \bar{p}$ and $\mathbf{u}(x, y - 1, t) \equiv \mathbf{0}$ for pressure and velocity BCs respectively.

Bounce-back (BB) rule

For a non-slip boundary, the BB rule sets an inward pointing distribution function $f_i(\mathbf{x}, t + \Delta t)$ equal to post-collision value pertaining to the opposite direction [62]:

$$f_i(\mathbf{x}, t + \Delta t) = f_{\bar{i}}(\mathbf{x}, t), \quad (6.20)$$

where $\mathbf{e}_{\bar{i}} = -\mathbf{e}_i$. Hence, the bounce-back method simulates the advection of a particle (population) which moves along the lattice direction $-\mathbf{e}_i$, bounces against a no-slip wall and is reflected back. Albeit a viscosity-dependent slip velocity, the BB rule is often employed to impose a zero-velocity at the lattice sites close to a complex non-slip wall. In the LBM, each particle travels a unitary distance at each time step. Therefore, the reflected particle returns to position \mathbf{x} if the wall is at half way through the lattice vector corresponding to direction i ($q_u = 1/2$ in Fig. 6.2). In effect, when this is the case the BB method is second-order accurate in space while in the other configurations it becomes first-order accurate [62], as confirmed by our results (see below). As discussed in Chapter 3 and shown by Junk and Yang [170], several velocity BCs reduce to the BB for the particular favourable boundary configuration cited

earlier; hence, we thought it was important to compare the accuracy results given by new or existing methods with that attained by the BB rule whilst adhering to that specific situation which is very simple to approach.

Yang's boundary condition method

The BCM recently presented by Yang [171] handles pressure-controlled lattice sites. The core idea was enriched with various rules to approach all the geometrical configurations which can occur at any pressure boundary topology. We did not implement and explore all the facets of the method but we approached this for an axis-aligned pressure boundary only. In this case, the inward pointing distribution function at time $t + \Delta t$ and boundary position \mathbf{x} is calculated as follows:

$$f_i(\mathbf{x}, t + \Delta t) = -f_{\bar{i}}(\mathbf{x}, t + \Delta t) + f_i(\mathbf{x}, t) + f_{\bar{i}}(\mathbf{x}, t) + 2w_i(\bar{\rho}(\mathbf{x}, t) - \rho(\mathbf{x}, t)), \quad (6.21)$$

where $\mathbf{e}_{\bar{i}} = -\mathbf{e}_i$, $\bar{\rho}(\mathbf{x}, t)$ is the prescribed density and w_i depends on the LB model (see Chapters 2 and 3). Yang proved that the method is second-order accurate in space for the velocity and first-order accurate for the pressure, but our results do not exhibit such a behaviour unless the Mach number is extremely low.

Accuracy of Results

The simulation results connected to the BCMs reviewed above and described in Chapter 3 are reported here. The results of Sec. 6.1.1 show that the LBM is intrinsically very accurate within a large range of spatial and temporal resolutions if the Mach number is ≈ 0.1 or below. Hence, the intrinsic spatial accuracy of various BCMs as a function of spatial resolution at sufficiently low Reynolds and Mach numbers is discussed. Then, some results related to $Re \gg 1$ are provided. The fluid flows were simulated within square ducts and cylinders.

First, we provide the results concerning the simulation of fluid flowing within a square duct with dimensions $(L_x, L_y, L_z) = (L_x, L_x, L_z)$ and whose boundaries are perpendicular to the Cartesian axes and driven by a pressure gradient $-\Delta P$ directed along axis Z. The analytical description is given by the following formulae (see Ref. [254]):

$$\frac{\partial p}{\partial z} = -\Delta P, \quad \frac{\partial p}{\partial x} = 0, \quad \frac{\partial p}{\partial y} = 0, \quad (6.22)$$

$$u_z(x, y, t) = \frac{64\Delta P}{\nu\pi^4} \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} g(x, y), \quad (6.23)$$

$$u_x(x, y, t) = u_y(x, y, t) = 0, \quad (6.24)$$

$$p(z) = p_i - \frac{2z + L_z}{4} \Delta P, \quad (6.25)$$

BCM		$\Delta x = 1/4$	$\Delta x = 1/8$	$\Delta x = 1/16$	$\Delta x = 1/32$	order
Guo <i>et al.</i> [6]	ξ_ρ	3.53e-1	1.72e-1	1.91e-1	2.81e-1	8.36e-2
	ξ_u	9.42e-1	4.84e-1	3.18e-1	3.94e-1	0.44
	ts	75	611	2936	10102	
BB + Zou and He [253]	ξ_ρ	1.07e-1	6.76e-2	2.80e-2	9.41e-3	1.18
	ξ_u	5.34e-1	1.77e-1	4.70e-2	1.18e-2	1.84
	ts	815	3345	14155	57116	
BB + Yang [171]	ξ_ρ	1.04e-1	8.97e-2	6.59e-2	5.36e-2	0.331
	ξ_u	5.51e-1	2.14e-1	9.91e-2	7.12e-2	0.997
	ts	933	2178	8221	30491	
BCM_{HemeLB}	ξ_ρ	1.03e-2	1.16e-2	1.44e-2	2.85e-2	-0.472
	ξ_u	6.04e-1	3.53e-1	1.68e-1	7.23e-2	1.03
	ts	108	513	2025	7085	
BCM^+_{HemeLB}	ξ_ρ	1.12e-2	1.92e-2	1.90e-2	1.40e-2	-9.51e-2
	ξ_u	1.22e-1	7.01e-2	3.70e-2	2.09e-2	0.856
	ts	190	634	2168	7148	

Table 6.5: Errors, orders of accuracy and number of time steps (ts) for the fluid flow within a square duct and simulated with the LBGK model coined D3Q15, $Re = 0.754$, maximal velocity $u_{max} = 0.03772\Delta x$ and $\nu = 0.05$. See text for further details.

where

$$g(x, y) = \frac{(-1)^{m+n}}{(2m+1)(2n+1)} \cos \frac{(2m+1)\pi x}{L_x} \cos \frac{(2n+1)\pi y}{L_y}, \quad (6.26)$$

for u_x , u_y and u_z are the velocities along axes X, Y and Z respectively, $p(z)$ is the pressure along the plane perpendicular to the axis Z and at position z , $p_i = c_s^2 + \Delta P \frac{L_z}{2}$ that at the inlet.

The convergence criterion and the error computation in the velocity and pressure fields are those described in Sec 6.1.1 which were modified to accommodate the third dimension. For the three-dimensional results we set $Tol = 10^{-6}$.

We set $q_u = 1$ and $q_p = 1$ (see Fig. 6.2) whilst employing the BCM described by Guo *et al.* [6]. The BB method was used in conjunction with the pressure BCM proposed by Zou and He [6] and that presented in Yang [171]; in these cases, we set $q_u = 1/2$ and $q_p = 0$ so as to maximise accuracy (see previous discussions). The BCMs presented in Chapter 3 and coined BCM_{HemeLB} and BCM^+_{HemeLB} were employed with $q_u = 0$ and $q_p = 0$.

Accuracy results for $Re = 0.753$, $\nu = 0.05$ and maximal velocity $u_{max} = 0.03772\Delta x$ (the lattice resolution is $\Delta x = 1/L_x$, $L_z = 2L_x = 2L_y$) are given in Table: 6.5.

We note that for $\Delta x = 1/16$ the most accurate simulation was achieved with

BCM		$\Delta x = 1/4$	$\Delta x = 1/8$	$\Delta x = 1/16$	$\Delta x = 1/32$	order
Guo <i>et al.</i> [6]	ξ_ρ	3.54e-1	3.15e-1	1.04e-1	9.25e-2	0.741
	ξ_u	9.39e-1	5.99e-1	1.98e-1	1.93e-1	0.884
	ts	933	2178	8221	30491	
BCM_{HemeLB}	ξ_ρ	3.14e-1	1.06e-1	5.05e-2	4.03e-2	0.996
	ξ_u	9.96e-1	8.79e-1	5.78e-1	3.17e-1	0.556
	ts	108	513	2025	7085	
BCM^+_{HemeLB}	ξ_ρ	2.79e-1	1.39e-1	6.73e-2	8.19e-2	0.635
	ξ_u	9.35e-1	2.84e-1	1.23e-1	1.06e-1	1.06
	ts	190	634	2168	7148	

Table 6.6: Errors, orders of accuracy and number of time steps (*ts*) for the fluid flow within a cylinder inclined by polar and vertical angles $\theta = \pi/3$ and $\phi = 2\pi/9$ respectively and simulated with the LBGK model coined D3Q15, $Re = 0.64$, maximal velocity $u_{max} = 0.032\Delta x$ and $\nu = 0.05$. See text for further details.

BCM^+_{HemeLB} while for the highest resolution the best accuracy was obtained with the BB method plus the BCM of Ref. [253]. The accuracy behaviour of the BCM presented by Guo *et al.* [6] appears very poor at any resolution. The BCMS BCM_{HemeLB} and BCM^+_{HemeLB} are of a zero-order accuracy for the pressure and of a first-order accuracy for the velocity; notably, for $\Delta x = 1/32$ BCM_{HemeLB} is more accurate than the BB method plus that developed by Yang [171]. It is worth noting that BB-based simulations need a number of iterations to converge an order of magnitude higher than that related to BCM_{HemeLB} and BCM^+_{HemeLB} .

Now, we present the simulation results of fluid flow within a cylinder of radius R and driven by the pressure gradient

$$\Delta P = \frac{4\nu u_0}{R^2}, \quad (6.27)$$

where u_0 is the centerline (peak) velocity. The formulae of the analytic pressure and velocity fields and of the convergence criterion are similar to those given in Sec. 6.1.1. As for the square duct, $Tol = 10^{-6}$.

The results in Table 6.6 were obtained by rotating the cylinder of $\phi = 2\pi/9$ rad along the Y axis and $\theta = \pi/3$ along the X axis so as to have different boundary-lattice intersections. In practise, q_u and q_p can be any. $Re = 0.64$, $\nu = 0.05$ and the maximal velocity is $0.032\Delta x$ ($\Delta x = \frac{1}{2R}$).

The new methods are competitive or superior to the BCM proposed by Guo *et al.* [6]. However, the accuracy is much lower than that achieved in the two-dimensional tests for similar or even higher Mach numbers. This means that, here, the error is dominated by that connected to the BCM –not to the LBM. High Reynolds number

cannot be achieved easily by adopting these BCs. For example, the simulation with $\Delta x = 1/32$, $u_{max} = 0.0133$, $Re = 114$ and $\nu = 0.00375$ is stable by only using BCM_{HemeLB} where $\xi_\rho = 0.379$ and $\xi_u = 0.385$. This means that we should wait for implementing more advanced BCs into our parallel tool HemeLB but Mazzeo and Gary Doctors are exploring various alternatives in order to increase the accuracy of three-dimensional fluid flow simulations in complex geometries.

6.2 Patient-specific cerebral blood flow simulations

In this section, three computational studies regarding blood flow modelling of intracranial aneurysms are presented in detail. Several works focussed on aneurysms were reviewed in Chapter 1. The intention of our investigations is to assess the efficacy of our computational tools (see Chapter 5 for their overview) in the haemodynamic characterisation of subject-specific vascular pathologies. In particular, we aim at verifying the presence of haemodynamic phenomena whose correlation to mechanisms, *e.g.* wall remodelling, has been recently demonstrated by other research groups. Thus, we seek to qualitatively evaluate aneurysm rupture risk on the basis of blood flow behaviour, which is valuable in the context of medical treatment planning.

We emulate the haemodynamics of each pathology at different spatio-temporal resolutions for numerical convergence purposes. In the near future, we plan to ascertain the impact of our software applications in real clinical scenarios, such as pre-operative planning and intra-surgical contexts. As discussed in Chapter 1, simulations make this possible by performing virtual surgery and by examining in advance the effect of particular treatment courses. We emphasize the importance of achieving accurate haemodynamic results in a short time to meet intra-surgical planning requirements. We will show that we can simulate the blood flow within a highly-resolved aneurysm model in minutes by conglomerating sufficient computational power.

The patient-specific volumetric datasets were obtained by means of the three-dimensional Rotational Angiography Imaging (3DRA) technique (see Chapter 1 and below for further details). Their three-dimensional reconstruction and boundary condition setup, needed for blood flow simulation, are carried out with an advanced in-house graphical-editing tool. The corresponding blood flow models were simulated and rendered at different spatial and temporal resolutions through HemeLB and its *in situ* visualisation approach. The results were analysed to determine blood flow behaviour in the patient-specific geometries and, in particular, in areas of the vasculatures potentially subjected to wall rupture. They indicate that some regions of the aneurysms are affected by a disordered velocity pattern and a very low stress, and consequently are prone to extensive wall remodelling, growth and rupture (see Chapter 1). We are currently exploring the use of more sophisticated segmentation

techniques and implementing more advanced BCMs in order to increase the accuracy of the segmentation process and of the fluid solver.

6.2.1 Patients, images and vascular models

We modelled three patient-specific aneurysms of different size and positioned in various locations within the cerebral vasculature. All volumetric data were obtained in DICOM format with a 3DRA scanner (see Chapter 1), specifically a Siemens Artis Zee [251] installed in the new angiography suite at the National Hospital for Neurology and Neurosurgery (NHNN). The isotropic resolution is 0.449 mm and comprises about 400 slices of 512^2 pixels each while every pixel is represented by 16 bit grey-scale. As discussed in Chapter 1, blood flow simulations are sensitive to geometrical variations in the shape of the segmented vasculature. In this context, we can conclude that our simulation results cannot be quantitative or very reliable since the diameter of the vessels and the size of the aneurysms considered here is only 3 – 20 times the pixel resolution. However, Geers *et al.* [15] show that qualitative information can be extracted even if the spatial resolution is about two times coarser than ours. Furthermore, we aim at using sub-pixel image processing to enhance segmentation accuracy and at developing more accurate BCMs for our LB solver.

In this work, the three vascular models were reconstructed with an efficient in-house graphical-editing tool (see Fig. 6.3). Each vasculature can be segmented and manipulated within one minute by the following process. The software takes the patient-specific dataset as input, and can interactively select and visualise the corresponding two-dimensional slices. The mouse serves to pick up a pixel of the vasculature after which it is quickly segmented; this stage entails knowing where to find a point of interest of the vasculature. However, all the human interventions are aided by the interactive capabilities of the tool, which quickly obviate errors occurred in any user intervention. The resolution of the medical dataset is not adequate for simulation purposes; consequently, it is increased to a user-selected value, typically between three and five times along each axis. The grey intensity at the spatial locations corresponding to the voxels of the computational grid which are in between the pixels of the original medical dataset are obtained by means of trilinear interpolation.

A standard threshold-based segmentation technique [252] is used to extract the vasculature, permitting the adjustment of the three-dimensional model in real-time with minimal memory consumption thanks to hierarchical data structures and an efficient clustering algorithm. Furthermore, several keyboard and mouse actions permit the user to interactively change the threshold, the corresponding reconstruction and visual parameters, as well as to select a different slice, edit, delete and create boundaries with ease. Specifically, the latter are generated perpendicular to the mouse-location-defined vessel and the mouse-based definition of inflow/outflow condi-

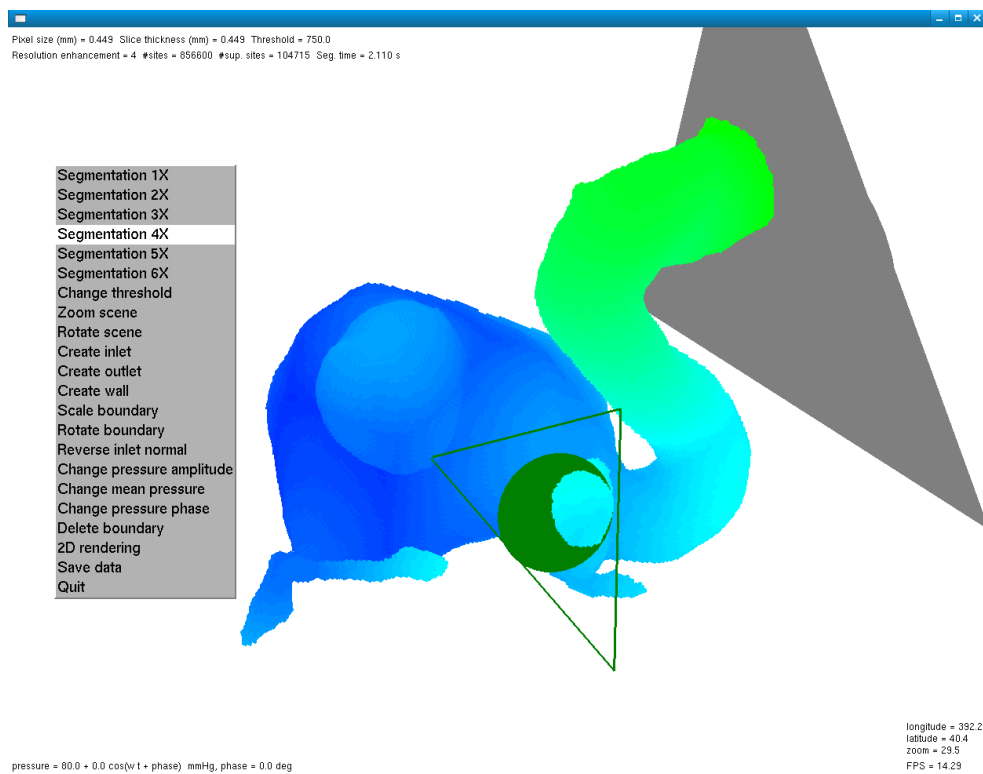


Figure 6.3: Snapshot of the graphical editing tool used to segment vasculatures and configure inflow/outflow boundary conditions. Several functions, as listed on the menu on the left hand side, are employed to configure boundary geometry and parameters. Here, we can see two boundaries (grey and green triangles) and the cropped vasculature therein.

<i>Model</i>	M_1	M_2	M_3
pathology	aneurysm	aneurysm	aneurysm
location	right MCA	left ICA	right MCA
size	large	large	small

Table 6.7: Geometrical information of the vascular patient-specific models investigated in this work; “small” and “large” mean about 5 mm and slightly more than 10 mm respectively. MCA and ICA stands for middle cerebral artery and internal carotid artery respectively.

tions, together with the information pertaining to the fluid voxels, serve as input for the blood flow simulator HemeLB (see Chapter 3). Table 6.7 lists some geometrical details of the three patient-specific vascular pathologies.

Threshold-based segmentation techniques can be subjected to considerable errors especially when the user is not an expert of vascular systems. We have recently enabled our graphical-editing software to import the segmentation results output by another tool thereby shifting the segmentation task to more accurate open-source or commercial tools.

6.2.2 Blood flow modelling and visualisation

Blood flow was modelled using HemeLB and all numerical or visual results were obtained through its parallel and grid-based *in situ* capabilities (see Chapters 3, 4 and 5). The simulations were monitored and steered in real-time using the GUI described in Chapter 5 while the simulation was running on the NGS-2 node [240] located at Leeds, UK; HARC was used to co-reserve the corresponding slots (see Chapter 5). We have also employed the supercomputing infrastructure Ranger of the TeraGrid [241] to demonstrate that we can accomplish the simulation of a highly-resolved cerebral vasculature in minutes.

Simulation setup

We have observed that the blood flow simulations in complex geometries with the LBM are prone to become numerically unstable, especially when the vasculature is highly-resolved. The origin of this is simple to understand since, in this case, the distance in lattice points between the boundaries is large and a high pressure gradient, which is traditionally at the origin of the numerical instability of the LBM [62], must be imposed to drive the fluid. Additionally, the viscosity is very low and this tends to yield sudden high spatial pressure gradients, in particular in the early stage of the simulation where the flow field rapidly passes from the arbitrary initial setup to the flow distribution that is close to the fully developed one.

If the vasculature is highly-resolved, several time steps are needed to transmit information between inflows and outflows, and generally any part of the system. We point out that a similar problem occurs when using other CFD techniques based on the direct discretisation of the Navier-Stokes equations: the pressure field calculated at every time step to guarantee fluid incompressibility entails solving a linear system with an iterative approach, that is repeatedly propagating any information between nearest-neighbour grid points, which requires many more steps to converge when the system is more complex and sparsely distributed [36]. We select a reasonable number of time steps for a specific spatial resolution and restart the simulation, doubling the number of time steps if an instability occurs. This overcomes the use of a temporal resolution which is too low for the current model, at the expense of a little increase in computational time². We tried to adjust viscosity and boundary conditions during the first pulsatile cycle to reduce instability incidence. For example, viscosity can be smoothly varied in time from a high value to the user-defined one. Analogously, the pressure conditions can be adjusted to begin with a zero pressure gradient and slowly adjust them to reach the nominal values at the end of the first pulsatile cycle. We experimented with these strategies but they did not noticeably reduce the occurrence of instability. However, we chose to start the simulation at a certain time during the diastole so as to flatten or minimise the pressure distribution (see text below).

Blood was assumed to have a constant density $\rho = 1000 \text{ kg/m}^3$ and dynamic viscosity $\mu = 0.004 \text{ Pa}\cdot\text{s}$. The inflow condition was a sinusoidal pressure pulse varying between 90 and 96 mm·Hg for model M_1 and between 90 and 100 for model M_2 and M_3 . 70 cardiac cycles were assumed to take place in a minute. At the outlet boundaries, a constant pressure equal to the minimum pressure attained at the inlets is imposed.

The boundary configurations were selected with care. The location of the outlets with respect to the inlet are set by assuming that the corresponding distal vascular beds have a similar total resistance to flow so that the decision to fix the outlets at the same pressure is reasonable. This rule of thumb takes into account the length of the vascular route between the inlet and the outlet, and the corresponding curvature profile.

Furthermore, the boundaries were set in relatively straight sections so as to avoid biasing the simulation results by the simplified boundary conditions, as discussed by Hassan *et al.* [47], which is important. The flow field was always initiated by setting all the distribution functions equal to the equilibrium ones calculated with zero velocity

²As shown below, a simulation takes about four pulsatile cycles to numerically converge and to make the influence of the initial conditions negligible. As cited earlier, instability usually appears in the early stage of the first pulsatile cycle. In this case, the corresponding number of time steps and computational time are modest with respect to those needed to perform the total simulation at twice the temporal resolution.

and outlet pressure (equal for each outflow boundary). In our set up, this causes the pressure gradient between any neighbouring lattice sites to vanish, and therefore substantially prevented numerical instability at the beginning of the simulation (see discussion at the beginning of this subsection).

The conversion between physical and lattice units is easily obtained by considering that the Reynolds and Womersley numbers of a particular fluid flow behaviour do not depend on the current system or resolution in time and space; hence, $\text{Re}^* = u_{max}^* D^* / \nu^* = \text{Re} = u_{max} D / \nu$ and $\alpha^* = \frac{D^*}{2} \sqrt{2\pi / (T^* \nu^*)} = \alpha = \frac{D}{2} \sqrt{2\pi / (T \nu)}$, where D is the characteristic length scale of the system and the symbol “*” denotes a parameter in physical units, assuming that the characteristic velocity $u \propto \nabla p \frac{D^2}{\rho \nu}$ in both lattice and physical units³:

$$\frac{u}{u^*} = \frac{D}{D^*} \frac{T^*}{T}, \quad (6.28)$$

$$\frac{\nu}{\nu^*} = \left(\frac{D}{D^*} \right)^2 \frac{T^*}{T}, \quad (6.29)$$

$$\nabla p = \nabla p^* \left(\frac{\rho D}{\rho^* D^*} \right) \left(\frac{T^*}{T} \right)^2, \quad (6.30)$$

where D is interchangeable with the resolution of the computational lattice Δx .

Numerical analysis

Each model was simulated at different spatial and temporal resolutions. Several haemodynamic and numerical properties were analysed through a number of parallel algorithms whose calculations run concurrently with HemeLB. The convergence criterion is the three-dimensional version of Eq. 6.13 with $Tol = 10^{-6}$. T is the number of time steps needed to complete a cardiac cycle. Number of fluid lattice sites (voxels), number of cardiac cycles to achieve numerical convergence (cycles), maximum velocity magnitude v_{max} and von Mises stress (see Chapter 2, Eqn. 2.20) s_{max} as a function of spatial resolution ΔX (mm) and T are reported in Table 6.8 where $\Delta X_k = 0.449/k$ mm for $k = 1, 2, 3$ and the corresponding T_k are $25K$, $50K$ and $100K$ for M_1 , $50K$, $50K$ and $100K$ for M_2 , and $25K$, $50K$ and $50K$ for M_3 . For brevity, Res_k stands for the spatio-temporal resolution $(\Delta X_k, T_k)$. These spatial and temporal resolutions were chosen in order to achieve numerically stable results whilst maintaining the computational time reasonable. In Table 6.8, we also report the average $\langle v_i \rangle$ and peak v_i^{max} inflow velocity magnitudes calculated by considering the component of the velocity parallel to the inflow boundary normal of each inlet voxel at every time step of the last pulsatile cycle.

³Eq. 6.30 holds for fluid flowing through a cylindrical channel with a circular section (Poiseuille flow) or a rectangular one (see Eq. 6.22), and we assume it to be valid in general.

Resolution	Res_1			Res_2			Res_3		
Model	M_1	M_2	M_3	M_1	M_2	M_3	M_1	M_2	M_3
voxels	126K	361K	54K	298K	857K	128K	581K	1670K	249K
cycles	4	6	3	4	6	3	5	6	3
v_{max}	1.14	1.53	1.21	0.996	1.42	1.30	1.18	1.45	1.191
s_{max}	67.5	84.63	70.1	51.3	87.7	90.2	88.2	101	91.3
$\langle v_i \rangle$	0.132	0.0913	0.0316	0.0895	0.105	0.0568	0.160	0.127	0.0673
v_i^{max}	0.682	0.633	0.237	0.496	0.546	0.406	0.856	0.918	0.297

Table 6.8: Blood flow results pertaining to the vascular patient-specific aneurysms M_1 , M_2 and M_3 investigated in this work. Velocity and stress values are expressed in m/s and Pa respectively. s_{max} and v_{max} are the maximum values of the von Mises stress (see Chapter 2, Eqn. 2.20) and of the velocity magnitude respectively, $\langle v_i \rangle$ and v_i^{max} are the average and peak inflow velocity magnitudes respectively. “voxels” and “cycles” are the number of fluid sites and cardiac cycles to achieve numerical convergence respectively. See text for further details.

For models M_1 and M_2 , the absolute maximum velocity magnitude double the maximum velocity at the inlet, while their ratio is between three and five for the last model. The maximum inlet velocity perpendicular to the inflow boundary is between 4.4 and 7.5 times its average value. Generally, the maximum stress increases as the spatio-temporal resolution becomes higher. The velocity flow field in terms of the estimated quantities is slightly faster by employing the highest resolution. This is probably due to the largest ratio of interior voxels over superficial ones which, approximating the surface of the vasculature in a staircase fashion (see Chapter 3), tend to significantly slow down blood motion. This is not well explored yet but we aim at circumventing it by incorporating more accurate and boundary-fitting schemes.

We have developed a post-processing tool which numerically compares the simulation results carried out at different spatial resolutions. The tool takes as input the fluid flow results carried out at the highest resolution and at the first simulation time step and stores it in a memory-cheap hierarchical grid. Then, it reads the results pertaining to the lowest resolution and, for each fluid lattice site, the software computes the difference between its flow field values (pressure, velocity and stress) and those stored in the hierarchical grid at the appropriate spatial location⁴.

Specifically, for each simulation the blood flow results at one hundred equally-spaced time steps were output by HemeLB. With this approach we can evaluate the average L_1 -error with respect to the blood flow model achieved with Res_3 taken as

⁴Sometimes, a fluid voxel at the lowest resolution does not have the spatial counterpart at the highest resolution and *viceversa*. In general, the geometrical similarity between two systems which differ in terms of resolution only, depends on the reconstruction stage.

Model	M_1		M_2		M_3	
	Res_1	Res_2	Res_1	Res_2	Res_1	Res_2
Pressure	0.166	0.0525	0.178	0.0953	0.542	0.234
Velocity	53.1	26.3	37.3	22.0	66.2	25.4
Stress	52.9	30.1	46.8	28.6	63.5	34.5

Table 6.9: Average L_1 -errors in percentage calculated by considering the difference in the pressure, velocity and von Mises stress flow fields of the three aneurysm patient-specific models M_1 , M_2 and M_3 of Table 6.7 carried out with the two lowest spatio-temporal resolutions compared with the results of the highest one. See text for further details.

reference. Its average value is reported in Table 6.9.

By observing Table 6.9, we note that the velocity and stress flow fields vary substantially between different spatio-temporal resolutions. This indicates that Res_2 is not appropriate to accurately study these complex geometries. As discussed in Chapter 1, small variations in the geometrical representation can lead to large differences in the simulation results. Specifically, if a path connects some parts of the model in a slightly different way which could depend on a poorly resolved representation of the vessel wall and therefore on the spatial resolution, the fluid flow behaviour may be affected by completely different haemodynamic features. This is an important aspect amplified by the presence of vortices which influence the haemodynamics of our models, as shown below.

Visual feedback

The *in situ* rendering technique can display different visual modalities. Here, we show the visual results obtained by employing the rendering choice discussed below. The parallel ray tracer (see Chapter 4) performs the volume rendering of the velocity and von Mises stress flow fields and calculates the wall pattern of the latter flow field and that of the pressure distribution. Each visual modality is performed in a very efficient single-pass technique, that is, a single ray is sent through a pixel of the viewport to accomplish all four rendering modes at once.

The simulation of model M_3 at the highest spatio-temporal resolution was also performed on 512 processor cores of the supercomputer Ranger of TeraGrid [241]⁵. It took approximately 9 minutes and ran at a rate of 287 time-steps per second by avoiding the convergence test (see Sec. 6.2.2), notwithstanding the continuous

⁵Ranger and the aforementioned NGS-2 node are composed by AMD Opteron processors, but Ranger has many more cores and its interconnect is considerably faster than the one installed on the NGS-2 node. Unfortunately, HARC cannot be used to co-reserve time on Ranger: the slots needed by us to conduct the interactive sessions on Ranger were reserved by its support team.

visualisation of results and steering of visual parameters at a highly interactive rate⁶. This demonstrates the efficacy of our simulation environment.

Blood flow motion is not properly perceived through volume rendering techniques. Massless time-dependent particles were continuously generated at random inlet positions and were allowed to follow the blood flow to form so-called streaklines [256], which substantially help to understand blood flow behaviour. Specifically, the velocity of each particle was obtained by tri-linearly interpolating the eight nearest-neighbour voxel values at the current particle location \mathbf{x} ; at each time step, the interpolated velocity $\mathbf{v}(t + \Delta t)$ was employed to update the particle position through a first order accurate symplectic (conservative) integration scheme: $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t + \Delta t)$. We note that the nearest neighbour voxels can reside outside the vasculature, where we assume there is zero velocity, or can be located in an adjacent processor subdomain; in this case, we perform data communication. The corresponding parallel (*in situ*) technique dynamically exchanges information between neighbouring processors whilst minimising the size of the messages to communicate. Specifically, the identifiers and the velocity of the voxels computed by neighbouring processors are communicated on demand; thus, we do not transmit all the interface-dependent information corresponding to the region between different sub-domains. Particle data are communicated in a similar way, and particles are created or deleted if they move across different sub-domains, or their velocity is below a user-defined value (10^{-4} lattice units in our case). The streaklines are displayed by superimposing particle visualisation, coloured accordingly to its velocity, onto the wall patterns in order to maximise visual feedback and scientific exploration.

The rendering snapshots of the three vascular models simulated at the highest spatio-temporal resolution and corresponding to the systolic and diastole peaks of the cardiac cycle are shown in Figs. 6.4, 6.5 and 6.6. The visual perception of the time-varying blood flow behaviour is not clear by statically displaying two time frames only. However, one can note that the fluid flow velocity and stress patterns at the systole is drastically different from that of the diastole. Wall stress at the body of M_1 and M_2 is very low. During the interactive session mouse-based events enabled the analysis of superficial pressure and stress, and confirmed that stress levels oscillated around 0.01 Pa at the tip of M_1 and M_2 . The streakline distribution is very disordered in those models. Moreover, M_2 and M_3 are characterised by a large pressure drop between the malformation and the outflow branches.

Videos comprising 100 images⁷ (written to file by HemeLB) clearly show other phenomena. Blood flow motion is very disordered in M_1 where vortices dynamically change their structure close to the surface of the aneurysm. The systolic phase pro-

⁶Notably, the simulation completes much more quickly if visual feedback is not provided at all times.

⁷These videos are provided at <http://ccs.chem.ucl.ac.uk/~marco/PatientSpecificSimulations/>.

vokes a sudden change in the blood circulation of M_2 . In the first stage of the pulsatile cycle the blood slowly flows through the route directly connecting the inlet and the outlet. Then, blood is rapidly injected into the aneurysm that provokes the formation of different vortices and a rapid increase in the stress distribution which extends towards the tip (impingement zone). Finally, the blood is quickly pumped through the outlet to leave a quasi-stationary aneurysm haemodynamics. The blood behaviour of M_3 is less disordered but a vortex starts to develop from the upper side of the malformation and propagates up to its neck close to the lower artery. Nonetheless, its structure changes slightly over time. Shojima *et al.* [53], Joua *et al.* [54] Bousset *et al.* [55] correlated aneurysm growth with low wall shear stress (see Chapter 1). In conclusion, typical conditions of aneurysms prone to experience further growth and rupture extensively characterise models M_1 and M_2 .

6.3 Conclusion

We have explored the intrinsic accuracy of the LBM in two dimensions. The corresponding results indicate that the LB can be very accurate in the simulation of steady and time-varying fluid flow within complex geometries. We also tested the accuracy of the LBM in three dimensions in conjunction with a few existing and new boundary condition methods. Their accuracy is poor compared to that attained in two dimensions which suggests that further investigation of velocity and pressure boundary conditions is required, especially for complex geometries.

We have validated the model used in HemeLB and a variant by simulating fluid flow in planar and curved boundaries. We plan to compare the simulation results of complex geometries obtained by HemeLB with those achieved with a well-validated fluid solver adopting the same conditions and parameters; this work is in progress. We can also compare computational results with those achieved with accurate *in vitro* measurements as accomplished by Boutsianis *et al.* [32]. This would be a more robust validation approach since it can also quantify the approximations introduced by assuming that the blood is a Newtonian fluid and that the vessel walls are rigid. *In vivo* velocity measurements are not impossible to obtain (see Chapter 1). We also plan to perform whole-brain blood flow simulations and to study arterio-venous malformations. Patient-specific boundary conditions were not available; we aim at acquiring pressure conditions through further collaboration with clinicians working at the National Hospital for Neurology and Neuroscience (NHNN), Queens Square, London. However, we plan to investigate the adoption of different segmentation techniques by using the open-source software ITK (Insight Segmentation and Registration Toolkit) [257] to reconstruct the vasculatures with a high fidelity.

We have demonstrated that we are able to simulate patient-specific haemodynam-

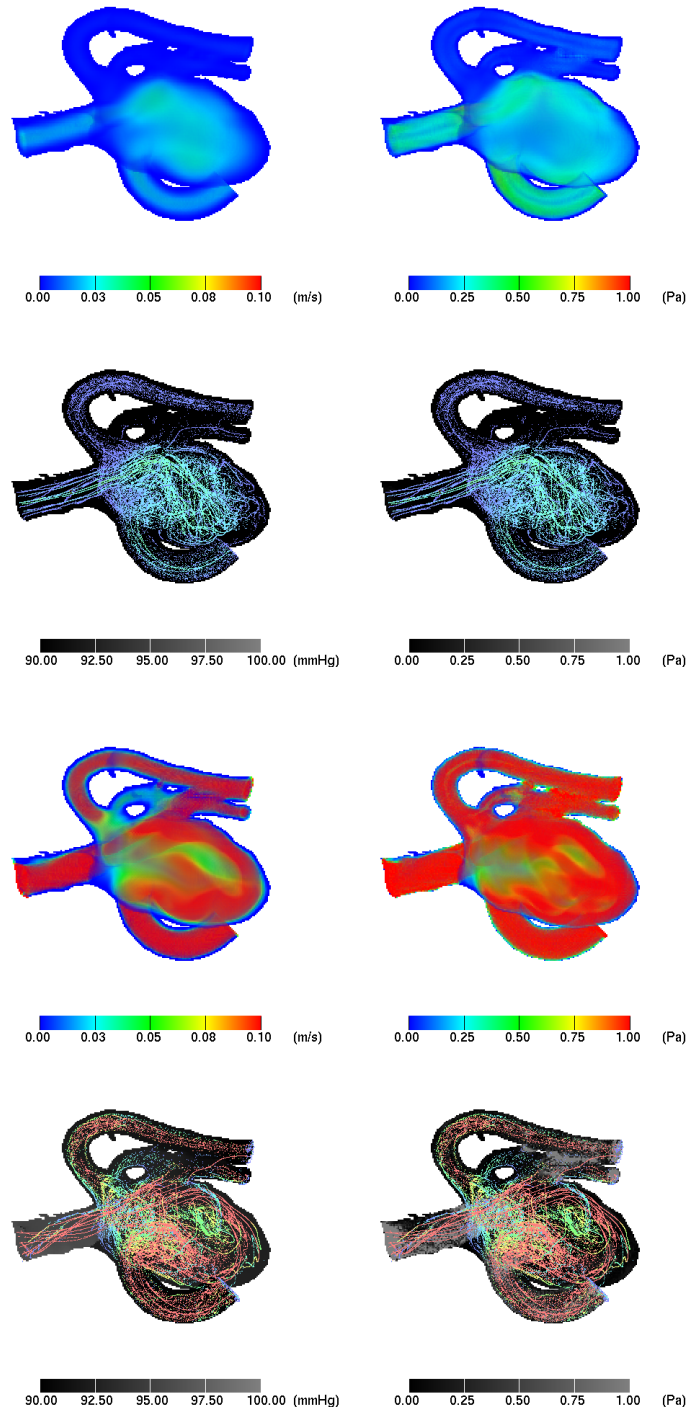


Figure 6.4: Eight snapshots subdivided in two panels with four quadrants each obtained at the diastolic (uppermost panel) and systolic (lowermost panel) pressure peaks of the aneurysm model M_1 obtained at the highest spatio-temporal resolution (see Sec. 6.2.2 for details). For each panel, the volume rendering of the velocity and von Mises stress flow fields are displayed within the top left and top right quadrants respectively; the bottom right and bottom left ones show the pressure and von Mises stress wall distributions, and particle traces (streaklines).

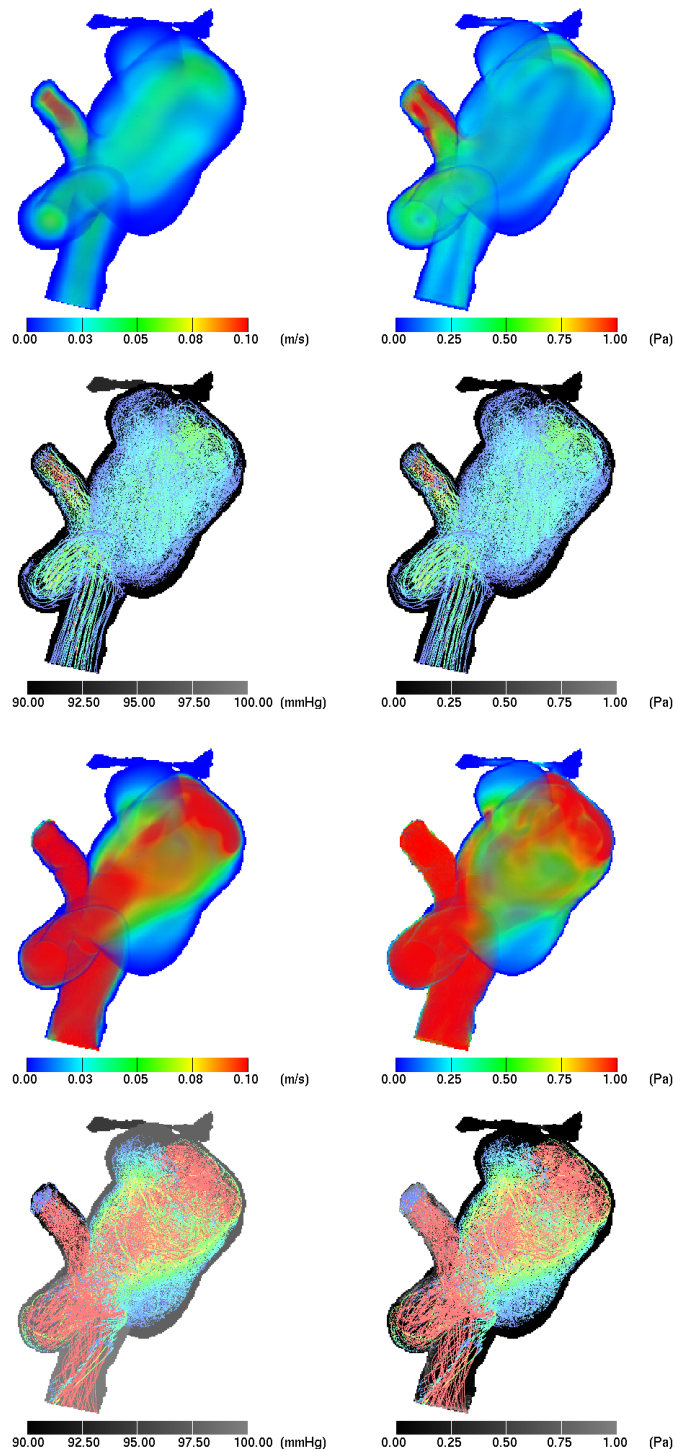


Figure 6.5: Eight snapshots subdivided in two panels with four quadrants each obtained at the diastolic (uppermost panel) and systolic (lowermost panel) pressure peaks of the aneurysm model M_2 obtained at the highest spatio-temporal resolution (see Sec. 6.2.2 for details). For each panel, the volume rendering of the velocity and von Mises stress flow fields are displayed at the top left and top right quadrants respectively; the bottom right and bottom left ones show the pressure and von Mises stress wall distributions, and particle traces (streaklines).

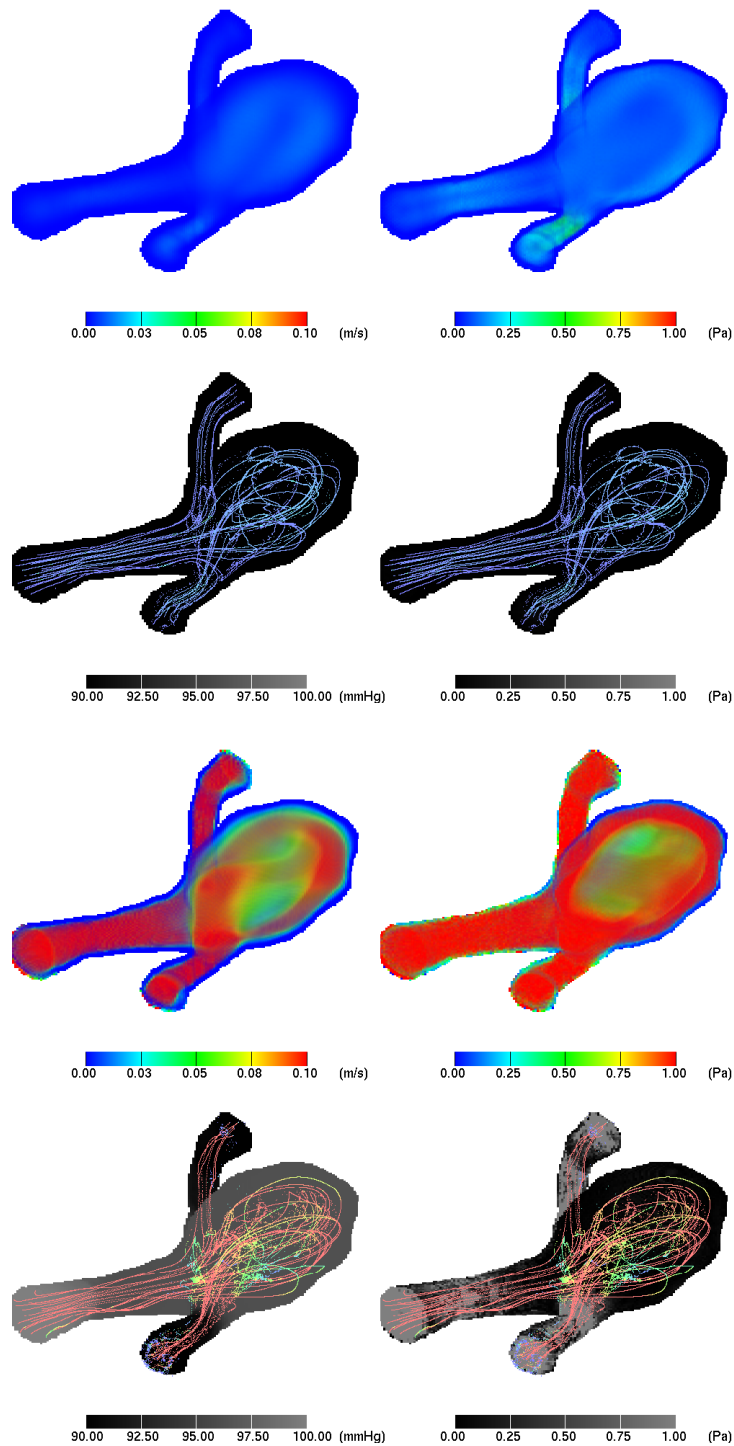


Figure 6.6: Eight snapshots subdivided in two panels with four quadrants each obtained at the diastolic (uppermost panel) and systolic (lowermost panel) pressure peaks of the aneurysm model M_3 obtained at the highest spatio-temporal resolution (see Sec. 6.2.2 for details). For each panel, the volume rendering of the velocity and von Mises stress flow fields are displayed at the top left and top right quadrants respectively; the bottom right and bottom left ones show the pressure and von Mises stress wall distributions, and particle traces (streaklines).

ics in real time, which gives a surgeon a powerful diagnostic tool to examine the effect of particular treatment courses in clinically relevant time scales. However, patient-specific medical simulation is a rather novel concept in medicine, and studies need to be developed and conducted which can give us insights into the verification and validation of such computational techniques, especially when used in real time. For example, by increasing accuracy in the image acquisition and segmentation processes, and in the blood flow simulation we may aim at giving powerful, reliable and quantitative information and in less time. For instance, our studies show that is necessary to discretise our systems with a very large number of fluid lattice sites because the fluid solver is not accurate; by using more accurate blood flow models and medical data, we can circumvent the aforementioned issue. Additionally, we may visualise the results in a way the clinicians are more familiar with.

Controlled clinical studies must be designed in which subgroups of patients would be recruited for treatment using current best practice, with and without computational simulation assistance, to assess the benefit to the patient. This issue of assessing “benefit” is not uncommon when new medical technologies are introduced. For example, a recent publication comparing the treatment of aneurysms by employing two-dimensional angiography and 3DRA was not able to make any firm conclusion about the advantage of using 3DRA [17]. However, in one particular case, where the vasculature surrounding an spinal aneurysm was sufficiently complicated, 3DRA gave the clinician a better understanding of the connective vascular structure. In conclusion, more studies are needed in order to confirm the benefit of the our software applications.

Conclusion

The highest result of education is tolerance

Helen Keller

Summary

This thesis began with a summary of concepts in the fields of haemodynamics and medical imaging techniques and reviewed numerous works of blood flow modelling carried out in the last fifteen years.

In Chapter 1, after a brief introduction of the main computational fluid dynamics techniques, we presented various theoretical concepts, shortcomings, capabilities and computational aspects of the lattice-Boltzmann method.

Then, the thesis described the lattice-Boltzmann fluid solver, called HemeLB, which was employed to obtain patient-specific blood flow results. Its computation and communication kernels are highly optimised and its development required a substantial amount of time over the last three years. The efficiency and scalability allow us to characterise haemodynamics of a highly-resolved patient-specific vasculature in a clinically relevant turnaround time frame if a sufficient computational power is employed. This is demonstrated in Chapter 6.

Chapter 4 presented our *in situ* interactive simulation and visualisation environment; its design and optimisation required a substantial effort and its capabilities permit us to visualise the simulation results in real time, and steer visual and simulation parameters from a commodity desktop while the blood flow is modelled and rendered by a supercomputer.

Between July 2007 and December 2008 part of our scientific activity was funded by the EPSRC project called GENIUS. Chapter 5 was focussed on the grid middleware developed within the GENIUS project; specifically, it permits one to easily control a series of software applications and facilities thanks to a simple graphical user interface. Specifically, the latter can control the tools to (a) book the computational resources in advance, which is very useful for interactive sessions, (b) support automated job launching, (c) reconstruct a patient-specific vasculature and (d) simulate, visualise and steer the corresponding blood flow field.

Finally, the thesis ended providing some blood flow results in rectilinear and patient-specific geometries. In particular, the lattice-Boltzmann method used in HemeLB recovers the analytical description of the time-dependent square duct fluid flow within acceptable accuracy, but currently only for low Re numbers. The haemodynamic studies qualitatively assessed the difference in accuracy between the simulation results obtained with different spatio-temporal resolutions, and demonstrated how, in the future, the model could potentially estimate the rupture risk of aneurysms. Furthermore, the efficacy of our software applications for image-based blood flow modelling was demonstrated.

Discussion and future work

Nowadays, the simulation tool plays an essential role in the modelling of blood flowing within complex vasculatures, but its accurate characterisation entails performing a large number of calculations. This can be accomplished by using a single processor for several hours and specialised software applications. Conversely, this work is the result of our aspiration to provide scientists and clinicians an user-friendly programming environment to approach this problem and as quickly as possible. The corresponding objective entailed handling a series of connected problems. For example, the use of grid computing is fundamental whenever an institution does not have the private access to large computational resources. Grid computing, in turn, entails employing stable and robust environments. Furthermore, the complexity of the resulting system complicates and lengthens some user tasks or even makes its employment impossible to non-expert users. The ambition to aid treatment planning during either pre-operative sessions or intra-surgical contexts pushed us to approach simulations on multiple machines, advance reservation of resources and urgent computing by augmenting existing collaborations or creating new ones. We have successfully addressed these issues. We have developed very high performance tools to preprocess medical images and model blood flow at impressive time scales. The rate at which our fluid solver processes data led us to approach the issue related to their effective visualisation and analysis. We overcame lengthy postprocessing stages by embedding parallel algorithms suitable for the calculation of several parameters and fast parallel visualisation methods with the fluid solver. We have also designed an optimised steering technique which permits one either to steer simulation parameters or to obtain the visual feedback returned by the fluid solver and its rendering components.

In Chapter 6, we mentioned some concepts related to the numerical instabilities which are particularly frequent during the simulation of complex highly-resolved vasculatures. However, we have spent a considerable time addressing this problem and we plan to diminish its impact.

We have also developed a number of useful programs to visualise, analyse and simulate two- and three-dimensional systems in different ways. These tools (not described in the thesis) were needed to study different boundary conditions, visualise different fluid flow patterns, improve performance and so on.

We can compare the simulation results obtained by means of HemeLB with those achieved through other fluid solvers; this important task will be accomplished soon to assess HemeLB accuracy in complex geometries. Giulia Di Tomaso and Vanessa Diaz (Dept. of Mechanical Engineering, University College London), can obtain the simulation results by employing the commercial software ANSYS CFX [258], while Greg Sheard (Monash University, Australia) can use a fluid solver based on the spectral element method [259].

In general, the accuracy of the vascular reconstruction is more important than that related to precise haemodynamic parameters, such as the precise inflow boundary condition [33–36]. For instance, too much smoothing in the segmentation process may mask genuine arterial morphology, while too little smoothing yields large spurious errors in wall stress patterns [260]. We need to better analyse the sensitivity of the simulation to the quality of the segmentation process. Specifically, we need to consider the employment of more accurate segmentation tools than that presented in Chapter 6. In this context, we will study the effect of the position of the inflow and outflow boundaries, because their approximate flow field setup, as inevitably occurs, affects the blood flow downstream.

We plan to implement non-Newtonian features and to study fluid flow confined by non-rigid walls by incorporating an elasticity model suitable for vessels subjected to small displacements, which is a reasonable assumption in non-large cerebral arteries (see Chapter 1). Once the related implementation has been validated, we will quantitatively verify the difference between the blood flow behaviour obtained with the corresponding LB code and the one which does not take into account elasticity and non-Newtonian effects.

We will also evaluate the application of inflow and outflow boundary conditions which are not determined solely by specific pressures as carried out in Chapter 6; for example, a resistance model where setting the pressure to be proportional to the flow rate is assumed to be adequate to model outflow at non-large vessels where pressure and velocity are in phase [25]; this would be suitable in situations where it is impossible to measure or estimate outlet pressures, as in an arterio-venous malformation.

We can also estimate other blood flow observables, *e.g.* the oscillatory shear index, the wall shear stress and its temporal variation, the vorticity of the flow in order to eventually better correlate haemodynamic patterns to critical areas of the patient's vasculature.

The interplay between blood flow dynamics and vessel wall mechanisms is essen-

tial to have important insights into the processes which lead to the formation and progression of vascular diseases. For instance, some of the publications discussed at the end of the first chapter provide evidence of the correlation between specific haemodynamic factors and vascular pathology progression, such as the effect of a very low wall shear stress on the wall remodelling and aneurysm growth. A long-term goal consists in considering and comparing the effects of different patient conditions, for instance rest versus exercise condition, symmetric versus asymmetric inflow conditions, and correlate haemodynamic observables to pathology formation or progression by means of the evaluation of physiological temporal changes in the patient vasculature, as carried out by Boussel *et al.* [55], which is very interesting and profitable. Collaboration with clinicians based at the National Hospital for Neurology and Neuroscience (Queen Square, London) will be invaluable for this scientific work as well as other future directions. For instance, a clinician can measure *in vivo* haemodynamic parameters whose employment during the simulation can lead to a highly accurate numerical subject-specific blood flow representation. Furthermore, by incorporating a computational approach based on the coupling between haemodynamic description and biomechanical wall responses, it will be possible to model the temporal change of the structure of a patient vasculature, and therefore to accurately predict pathology formation and progression; this will be possible if the computational method describing vessel mechanisms is based on experimental data.

Finally, by further collaborating with clinicians we can investigate patient-specific problematics on a regular basis to aid their making-decision process. For instance, the fluid solver will be used as a tool to model haemodynamic conditions in the presence of a real vasculature and a configuration similar to the post-interventional situation to assess the effectiveness of a certain treatment, such as coiling, embolisation or surgical intervention. Moreover, by performing whole-brain simulations, we will investigate the role of the Circle of Willis on the rest of the cerebral circulation, including malformed regions downstream, and the haemodynamic interplay between an anomaly in the Circle of Willis itself and the rest of the system.

All these points and future working directions still need to be addressed. Our computational infrastructure is a promising clinical toolkit but whether it is eligible to be used for clinical scenarios remains an open issue.

Bibliography

- [1] B. Hillen, T. Gaasbeek, and H. W. Hoogstraten. A mathematical model of the flow in the posterior communicating arteries. *J. Biomech.*, 15(6):441–448, 1982.
- [2] L. A. Bortolotto and M. E. Safar. Blood pressure profile along the arterial tree and genetics of hypertension. *Brazilian Archives of Cardiology*, 86(3):166–169, 2006.
- [3] R. J. Cezbral, M. A. Castro, J. E. Burgess, R. S. Pergolizzi, M. J. Sheridan, and C. M. Putman. Characterization of cerebral aneurysms for assessing risk of rupture by using patient-specific computational hemodynamics models. *Am. J. Neuroradiol.*, 26:2550–2559, 2005.
- [4] J. L. Brisman, J. K. Song, and D. W. Newell. Cerebral Aneurysms. *N. Engl. J. Med.*, 355(9):928–939, 2006.
- [5] J. Wang, X. Zhang, A. G. Bengough, and J. W. Crawford. Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media. *Phys. Rev. E*, 72(2):016706–016716, 2005.
- [6] Z.-L. Guo, C.-G. Zheng, and B.-C. Shi. Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice Boltzmann method. *Chinese Phys.*, 11(4):366–374, 2002.
- [7] World Health Organization, The World Health Report 2002, Reducing risks, promoting healthy life. <http://www.who.int/whr/en/>.
- [8] A. Quarteroni, M. Tuveri, and A. Veneziani. Computational Vascular Fluid Dynamics: Problems, Models and Methods. *Comput. Vis.. Sci.*, 2(4):163–197, 2000.
- [9] R. S. Rosenson, A. McCormick, and E. F. Uretz. Distribution of blood viscosity values and biochemical correlates in healthy adults. *Clin. Chem.*, 42(8):1189–1195, 1996.
- [10] B. Hillen, H. W. Hoogstraten, and L. Post. A mathematical model of the flow in the circle of Willis. *J. Biomech.*, 19(3):187–194, 1986.

- [11] B. Hillen, B. A. Drinkenburg, H. W. Hoogstraten, and L. Post. Analysis of flow and vascular resistance in a model of the circle of Willis. *J. Biomech.*, 21(10):807–814, 1988.
- [12] M. D. Ford, N. Alperin, S. H. Lee, D. W. Holdsworth, and D. A. Steinman. Characterization of volumetric flow rate waveforms in the normal internal carotid and vertebral arteries. *Physiological Measurement*, 26:477–488, 2005.
- [13] J. Alastruey, K. Parker, J. Peiró, S. Byrd, and S. Sherwin. Modelling the circle of Willis to assess the effects of anatomical variations and occlusions on cerebral flows. *J. Biomech.*, 40(8):1794–1805, 2007.
- [14] T. Hassan, E. V. Timofeev, M. Ezuraa, T. Saito, A. Takahashia, K. Takayamac, and T. Yoshimoto. Hemodynamic Analysis of an Adult Vein of Galen Aneurysm Malformation by Use of 3D Image-Based Computational Fluid Dynamics. *J. Biomech.*, 24(6):1075–1082, 2003.
- [15] A. J. Geers, I. Larrabide, H. G. Radaelli, H. Bogunovic, H. A. F. G. van Andel, C. B. Majoie, and A. F. Frangi. Reproducibility of image-based computational hemodynamics in intracranial aneurysms: Comparison of CTA and 3DRA. In *IEEE International Symposium on Biomedical Imaging*, pages 610–613, 2009.
- [16] D. A. Steinman. Image-based computational fluid dynamics modeling in realistic arterial geometries. *Ann. Biomed. Eng.*, 30:483–497, 2002.
- [17] L. Jiang, C.-G. Huangq, P. Liu, B. Yan, J.-X. Chen, H.-R. Chen, R.-L. Bai, and Y.-C. Lu. 3-Dimensional rotational angiography for the treatment of spinal cord vascular malformations. *Surg. Neurol.*, 69(4):369–373, 2008.
- [18] M. A. Castro, C. M. Putman, and J. R. Cebal. Patient-Specific Computational Fluid Dynamics Modeling of Anterior Communicating Artery Aneurysms: A Study of the Sensitivity of Intra-Aneurysmal Flow Patterns to Flow Conditions in the Carotid Arteries. *AJNR Am J Neuroradiol.*, 27(10):2061–2068, 2006.
- [19] S. Wetzels, S. Meckel, A. Frydrychowicz, L. Bonati, E.-W. Radue, K. Scheffler, J. Hennig, and M. Markl. In Vivo Assessment and Visualization of Intracranial Arterial Hemodynamics with Flow-Sensitized 4D MR Imaging at 3T. *AJNR Am J Neuroradiol.*, 28(3):433–438, 2007.
- [20] C. A. Taylor, M. T. Draney, J. P. Ku, D. Parker, B. N. Steele, K. Wang, and C. K. Zarins. Predictive Medicine: Computational Techniques in Therapeutic Decision-Making. *Computer Aided Surgery*, 4:231–247, 1999.
- [21] S. Tateshima, Y. Murayama, J. P. Villablanca, T. Morino, K. Nomura, K. Tanishita, and F. Vi nuela. In Vitro Measurement of Fluid-Induced Wall Shear

- Stress in Unruptured Cerebral Aneurysms Harboring Blebs. *Stroke*, 34(1):187–192, 2003.
- [22] A. Frydrychowicz, E. Weigang, M. Langer, and M. Markl. Flow-sensitive 3D magnetic resonance imaging reveals complex blood flow alterations in aortic Dacron graft repair. *Interact. CardioVasc. Thorac. Surg.*, 5:340–342, 2006.
- [23] S. Yamashita, H. Isoda, M. Hirano, H. Takeda, S. Inagawa, Y. Takehara, M. T. Alley, M. Markl, N. J. Pelc, and H. Sakahara. Visualization of hemodynamics in intracranial arteries using time-resolved three-dimensional phase-contrast MRI. *J. Magn. Reson. Imaging*, 25(3):473–478, 2007.
- [24] S. Meckel, A. F. Stalder, F. Santini, E.-W. Radü, D. A. Rüfenacht, M. Markl, and S. G. Wetzel. In vivo visualization and analysis of 3-D hemodynamics in cerebral aneurysms with flow-sensitized 4-D MR imaging at 3 T. *Neuroradiol.*, 50(6):473–484, 2008.
- [25] M. S. Olufsen. Structured tree outflow condition for blood flow in larger systemic arteries. *Am J. Physiol. Heart Circ. Physiol.*, 276:H257–H268, 1999.
- [26] K. Perktold and M. Hofer. *Mathematical Modelling of Flow Effects and Transport Processes in Arterial Bifurcation Models Medium In: Advances in Computational Bioengineering, The Haemodynamics of Arterial Organs - Comparison of Computational Predictions with In vivo and In vitro Data*. Edited by X.Y.Xu and M.W.Collins, WIT-Press - Computational Mechanics Publications, Southampton, Boston, 1999.
- [27] F. J. H. Gijssen, F. N. van de Vosse, and J. D. Janssen. The Influence of the Non-Newtonian Properties of Blood on the Flow in Large Arteries: Steady Flow in a Carotid Bifurcation Model. *J. Biomech.*, 32:601–608, 1999.
- [28] A. Leuprecht and K. Perktold. Computer Simulation of Non-Newtonian Effects on Blood Flow in Large Arteries. *Comput. Met. Biomech. Biomed. Eng.*, 4:149–163, 2001.
- [29] G. Pontrelli. Pulsatile blood flow in a pipe. *Comput. Fluids.*, 27(3):367–380, 1998.
- [30] S. Z. Zhao, X. Y. Xu, A. D. Hughes, S. A. Thom, A. V. Stanton, B. Ariff, and Q. Long. Blood flow and vessel mechanics in a physiologically realistic model of a human carotid arterial bifurcation. *J. Biomech.*, 33(8):975–984, 2000.
- [31] E. S. Di Martino, G. Guadagni, G. Ballerini A. Fumero, R. Spirito, P. Biglioli, and A. Redaelli. Fluid-structure interaction within realistic three-dimensional

- models of the aneurysmatic aorta as a guidance to assess the risk of rupture of the aneurysm. *Med. Eng. Phys.*, 23(9):647–655, 2001.
- [32] E. Boutsianis, M. Guala, Y. Olgac, S. Wildermuth, K. Hoyer, Y. Ventikos, and D. Poulidakos. CFD and PTV Steady Flow Investigation in an Anatomically Accurate Abdominal Aortic Aneurysm. *J. Biomech. Eng.*, 131(1), 2009.
- [33] J. G. Myers, J. A. Moore, M. Ojha, K. W. Johnston, and C. R. Ethier. Factors Influencing Blood Flow Patterns in the Human Right Coronary Artery. *Ann. Biomed. Eng.*, 29(2):109–120, 2001.
- [34] O. J. Ilegbusi and Z. Hu and R. Nesto and S. Waxman and D. Cyganski and J. Kilian and P. H. Stone and C. L. Feldman. Determination of blood flow and endothelial shear stress in human coronary artery in vivo. *J. Invasive Cardiol.*, 11(2):667–674, 1999.
- [35] M. R. Kaazempur-Mofrad and C. R. Ethier. Mass Transport in an Anatomically Realistic Human Right Coronary Artery. *Ann. Biomed. Eng.*, 29(2):121–127, 2003.
- [36] J. R. Cebal, M. A. Castro, S. Appanaboyina, C. M. Putman, D. Millan, and A. F. Frangi. Efficient Pipeline for Image-Based Patient-Specific Analysis of Cerebral Aneurysm Hemodynamics: Technique and Sensitivity. *IEEE Trans. Medical Imaging*, 24(4):457–467, 2005.
- [37] Y. Liu, Y. Lai, A. Nagaraj, B. Kane, and A. Hamilton. Pulsatile flow simulation in arterial vascular segments with intravascular ultrasound images. *Med. Eng. Phys.*, 23(8):583–595, 2001.
- [38] C. M. Gibson, L. Diaz, K. Kandarpa, F. M. Sacks, R. C. Pasternak, T. Sandor, C. Feldman, and P. H. Stone. Relation of vessel wall shear stress to atherosclerosis progression in human coronary arteries. *Arterioscler. Thromb. Vascul. Biol.*, 13:310–315, 1993.
- [39] Y. Hoi, S. H. Woodward, M. Kim, D. B. Taulbee, and H. Meng. Validation of CFD Simulations of Cerebral Aneurysms With Implication of Geometric Variations. *J. Biomech. Eng.*, 128(6):844–851, 2006.
- [40] J. Boyd, J. Buick, J. A. Cosgrove, and P. Stansell. Application of the lattice Boltzmann model to simulated stenosis growth in a two-dimensional carotid artery. *Phys. Med. Biol.*, 50(20):4783–5779, 2005.
- [41] J. Boyd and J. Buick. Three-dimensional modelling of the human carotid artery using the lattice Boltzmann method: I. Model and velocity analysis. *Phys. Med. Biol.*, 53(20):5767–4796, 2008.

- [42] S. Moore, T. David, J. Chase, J. Arnold, and J. Fink. 3D models of blood flow in the cerebral vasculature. *J. Biomech.*, 39:1454–1463, 2006.
- [43] R. Botnar, G. Rappitsch, M. B. Scheidegger, D. Liepsch, K. Perktold, and P. Boesiger. Hemodynamics in the carotid artery bifurcation: a comparison between numerical simulations and in vitro MRI measurements. *J. Biomech.*, 33(2):137–144, 2000.
- [44] H. Ujiie, H. Tachibana, O. Hiramatsu, A. L. Hazel, T. Matsumoto, Y. Ogasawara, H. Nakajima, T. Hori, K. Takakura, and F. Kajiya. Effects of Size and Shape (Aspect Ratio) on the Hemodynamics of Saccular Aneurysms: A Possible Index for Surgical Treatment of Intracranial Aneurysms. *J. Neurosurg.*, 45(1):119–130, 1999.
- [45] H. Meng, Z. Wang, Y. Hoi, L. Gao, E. Metaxa, D. D. Swartz, and J. Kolega. Complex Hemodynamics at the Apex of an Arterial Bifurcation Induces Vascular Remodeling Resembling Cerebral Aneurysm Initiation. *Am. J. Neuroradiol.*, 38:1924–1931, 2007.
- [46] J. R. Womersley. Method for the calculation of velocity, rate flow, and viscous drag in arteries when the pressure gradient is known. *J. Physiol.*, 127:553–563, 1955.
- [47] T. Hassan, E. V. Timofeev, T. Saito, , H. Shimizu, M. Ezura, T. Tominaga, A. Takahashi, and K. Takayama. Computational Replicas: Anatomic Reconstructions of Cerebral Vessels as Volume Numerical Grids at Three-Dimensional Angiography. *AJNR Am J Neuroradiol.*, 25(8):1356–1365, 2004.
- [48] J. S. Milner, J. A. Moore, B. K. Rutt, and D. A. Steinman. Hemodynamics of human carotid artery bifurcations: computational studies with models reconstructed from magnetic resonance imaging of normal subjects. *J. Vasc. Surg.*, 28(1):143–156, 1998.
- [49] P. Venugopal, D. Valentino, H. Schmitt, J. P. Villablanca, F. Vi nuela, and G. Duckwiler. Sensitivity of patient-specific numerical simulation of cerebral aneurysm hemodynamics to inflow boundary conditions. *J. Neurosurg.*, 106:1051–1060, 2007.
- [50] I. Chatziprodmou, D. Poulikakos, and Y. Ventikos. On the influence of variation in haemodynamic conditions on the generation and growth of cerebral aneurysms and atherogenesis: A computational model. *J. Biomech.*, 40(16):3626–3640, 2007.

- [51] A. Gnasso, C. Carallo, C. Irace, M. S. De Franceschi, P. L. Mattioli, C. Motti, and C. Cortese. Association between wall shear stress and flow-mediated vasodilation in healthy men. *Atheroscler.*, 156(1):171–176, 2001.
- [52] D. N. Ku, D. P. Giddens, C. Z. Zarins, and S. Glagov. Pulsatile flow and atherosclerosis in the human carotid bifurcation. *Arterioscler.*, 5(8):293–302, 1985.
- [53] M. Shojima, M. Oshima, K. Takagi, R. Torii, M. Hayakawa, K. Katada, A. Morita, and T. Kirino. Magnitude and role of wall shear stress on cerebral aneurysm: Computational fluid dynamic study of 20 middle cerebral artery aneurysms. *Stroke*, 35(11):2500–2505, 2004.
- [54] L.-D. Jou, G. Wong, B. Dispensac, M. T. Lawton, R. T. Higashida, W. L. Young, and D. Saloner. Correlation between luminal geometry changes and hemodynamics in fusiform intracranial aneurysms. *Arterioscler.*, 26(9):2357–2363, 2005.
- [55] L. Boussel, V. Rayz, C. McCulloch, A. Martin, G. Acevedo-Bolton, M. Lawton, R. Higashida, W. S. Smith, W. L. Young, and D. Saloner. Aneurysm Growth Occurs at Region of Low Wall Shear Stress. Patient-Specific Correlation of Hemodynamics and Growth in a Longitudinal Study. *Stroke*, 2008.
- [56] Aneurist, integrated Biomedical informatics for the management of cerebral aneurysms. <http://www.cilab.upf.edu/aneurist1/>.
- [57] J. Tu, G. H. Yeoh, and C. Liu. *Computer Architecture: A Quantitative Approach, 2nd Edition*. Butterworth-Heinemann, 2007.
- [58] J. Blazek. *Computational fluid dynamics: principles and applications*. Elsevier, 2001.
- [59] S. F. Li and W. K. Liu. Meshfree and particle methods and their applications. *Applied Mechanics Review*, 55:1–34, 2002.
- [60] P. J. Hoogerbrugge and J. M. V. A. Koelman. Simulating microscopic hydrodynamics phenomena with dissipative particle dynamics. *Europhys. Letters*, 19:155–160, 1992.
- [61] A. Malevanets and R. Kapral. Continuous-velocity lattice-gas model for fluid flow. *Europhys. Letters*, 44:552–558, 1998.
- [62] S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.
- [63] S. Chen and D. Doolen. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364, 1998.

- [64] D. R. Noblem, J. G. Georgiadis, and R. O. Buchius. Comparison of Accuracy and Performance for Lattice Boltzmann and Finite Difference Simulations of Steady Viscous Flow. *Int. J. Num. Meth. Fluids*, 23(1):1–18, 1996.
- [65] K. Sankaranarayanan, I. G. Kevrekidis, S. Sundaresan, J. Lu, and G. Trygvason. A comparative study of lattice Boltzmann and front-tracking finite-difference methods for bubble simulations. *Int. J. Multiphase Flow*, 29(1):109–116, 2003.
- [66] M. Breuer, J. Bernsdorf, T. Zeiser, and F. Durst. Accurate computations of the laminar flow past a square cylinder based on two different methods: Lattice-Boltzmann and finite-volume. *Int. J. Heat Fluid Flow*, 21(2):186–196, 2000.
- [67] J. Bernsdorf, F. Durst, and M. Schäfer. Comparison of cellular automata and finite volume techniques for simulation of incompressible flows in complex geometries. *Int. J. Heat Fluid Flow*, 29(3):251–264, 2000.
- [68] X. He, G. Duckwiler, and D. J. Valentino. Lattice Boltzmann simulation of cerebral artery hemodynamics. *Comput. Fluids*, 38(4):789 – 796, 2009.
- [69] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on Lattice-Boltzmann, Finite Element and Finite Volume Methods for laminar Flows. Technical report, FB Mathematik, Universität Dortmund, 2004. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 274.
- [70] B. D. Kandhai, D. J.-E. Vidal, A. G. Hoekstra, H. C. J. Hoefsloot, P. Iedema, and P. M. A. Sloot. Lattice-Boltzmann and finite element simulations of fluid flow in a SMRX mixer. *Int. J. Numer. Meth. Fluids*, 31:1019–1033, 1999.
- [71] D. O. Martinez, W. H. Matthaeus, S. Chen, and D. C. Montgomery. Comparison of spectral method and lattice Boltzmann simulations of two-dimensional hydrodynamics. *Phys. Fluids*, 6:1285–1298, 1994.
- [72] S. Chen, Z. Wang, X. Shan, and G. D. Doolen. Lattice Boltzmann computational fluid dynamics in three dimensions. *J. Stat. Phys.*, 68:379–400, 1992.
- [73] T. Pohl, N. Thuerey, F. Deserno, U. Ruede, P. Lammers, G. Wellein, and T. Zeiser. Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*. IEEE/ACM SIGARCH, 2004.
- [74] A. M. M. Artoli, B. D. Kandhai, H. C. J. Hoefsloot, A. G. Hoekstra, and P. M. A. Sloot. Lattice Boltzmann, a Robust and Accurate Solver for Interactive Computational Hemodynamics. In P. M. A. Sloot, D. Abramson, A. V. Bogdanov,

- J. J. Dongarra, A. Y. Zomaya, and Y. E. Gorbachev, editors, *Computational Science - ICCS 2003, Melbourne, Australia and St. Petersburg, Russia, Proceedings Part I*, volume 2657 of *Lect. Notes Comput. Sci.*, pages 1034–1043. Springer Verlag, 2003.
- [75] M. Krafczyk, M. Cerrolaza, M. Sculz, and E. Rank. Analysis of 3D transient blood flow passing through an artificial aortic valve by Lattice-Boltzmann methods. *J. Biomech.*, 31:453–462, 1998.
- [76] M. Hirabayashi, M. Ohta, D. A. Rüfenacht, and B. Chopard. A lattice Boltzmann study of blood flow in stented aneurism. *Fut. Gen. Comp. Sys.*, 20(6):925–934, 2004.
- [77] H. W. Stockman. Effect of Anatomical Fine Structure on the Flow of Cerebrospinal Fluid in the Spinal Subarachnoid Space. *J. Biomech. Eng.*, 128:106–114, 2006.
- [78] H. Fang, Z. Wang, Z. Lin, and M. Liu. Lattice Boltzmann method for simulating the viscous flow in large distensible blood vessels. *Phys. Rev. E*, 65(5):051925.1–051925.11, 2002.
- [79] A. M. M. Artoli, A. G. Hoekstra, and P. M. A. Slood. Simulation of a Systolic Cycle in a Realistic Artery with the Lattice Boltzmann BGK Method. *Int. J. Modern Phys. B*, 17(1&2):95–98, 2003.
- [80] A. M. M. Artoli, B. D. Kandhai, H. C. J. Hoefsloot, A. G. Hoekstra, and P. M. A. Slood. Lattice BGK simulations of flow in a symmetric bifurcation. *Fut. Gen. Comp. Sys.*, 20(6):909–916, 2004.
- [81] H. Li, X. Lu, H. Fang, and Z. Lin. Simulation of multi-particle suspensions in a quasi-two-dimensional symmetric stenotic artery with lattice Boltzmann method. *Prog. Comput. Fluid Dyn.*, 5:65–74, 2005.
- [82] O. Pelliccioni, M. Cerrolaza, and R. Surós. A biofluid dynamic computer code using the general lattice boltzmann equation. *Adv. Eng. Softw.*, 39(7):593–611, 2008.
- [83] A. J. C. Ladd. Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation Part I. Theoretical Foundation. *J. Fluid. Mech.*, 271:285–309, 1993.
- [84] A. G. Hoekstra, J. van't Hoff, A. M. Artoli, and P. M. A. Slood. Unsteady flow in a 2D elastic tube with the LBGK method. *Fut. Gen. Comp. Syst*, 20(6):917–924, 2004.

- [85] D. d’Humières, P. Lallemand, and U. Frish. Lattice Gas Models for 3D Hydrodynamics. *Europhys. Lett.*, 2(4):291–297, 1986.
- [86] U. Frish, D. d’Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Syst.*, 1:649–707, 1987.
- [87] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models An Introduction*. Springer-Verlag, 2000.
- [88] G. R. McNamara and G. Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Phys. Rev. Lett.*, 61(20):2332–2335, 1988.
- [89] F. J. Higuera, S. Succi, and R. Benzi. Lattice gas dynamics with enhanced collisions. *Europhys. Lett.*, 9(4):345–349, 1989.
- [90] F. J. Higuera and S. Succi. Simulating the flow around a circular-Cylinder with a Lattice Boltzmann-Equation. *Europhys. Lett.*, 8:517–521, 1989.
- [91] L. Boltzmann. *Vorlesungen über Gastheorie*. Lectures on Gas Theory, Dover, 1995.
- [92] X. He and L.-S. Luo. A priori derivation of the lattice Boltzmann equation. *Phys. Rev. E*, 55(6):R6333–R6336, 1997.
- [93] X. He and L.-S. Luo. Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Phys. Rev. E*, 56(6):6811–6817, 1997.
- [94] T. Abe. Derivation of the lattice Boltzmann method by means of the discrete ordinate method for the Boltzmann equation. *J. Comput. Phys.*, 131(6):241–246, 1997.
- [95] M. Junk. A Finite Difference Interpretation of the Lattice Boltzmann Method. *Numer. Methods Partial Differ. Equations*, 17(4):383–402, 2001.
- [96] P. Lallemand and L.-S. Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. Technical Report TR-2000-17, Institute for Computer Applications in Science and Engineering, 2000.
- [97] L.-S. Luo. Analytic Solutions of Linearized Lattice Boltzmann Equation for Simple Flows. *Stat. Phys.*, 88:913–926, 1997.
- [98] D. d’Humières and P. Lallemand. Numerical simulations of hydrodynamics with lattice gas automata in two dimension. *Complex Syst.*, 1:599–632, 1987.

- [99] 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. *Phys. Rev.*, 94(3):511–525, 1954.
- [100] H. Chen, S. Chen, and W. H. Matthaeus. Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method. *Phys. Rev. A*, 45(8):R5339–R5342, 1992.
- [101] Y. Qian, D. d’Humières, , and P. Lallemand. Lattice BGK models for Navier-Stokes equation. *Europhys. Lett.*, 17(6):479–484, 1992.
- [102] P. J. Dellar. Incompressible limits of lattice Boltzmann equations using multiple relaxation times. *J. Comput. Phys.*, 190(6):351–370, 2003.
- [103] A. M. M. Artoli, A. G. Hoekstra, and P. M. A. Sloot. Optimizing lattice Boltzmann simulations for unsteady flows. *Comput. Fluids*, 35(2):227–240, 2006.
- [104] F. Toschi and S. Succi. Lattice Boltzmann method at finite-Knudsen numbers. *Europhys. Lett.*, 69(4):549–555, 2004.
- [105] Q. Zou, S. Hou, S. Chen, and G. D. Doolen. An improved incompressible lattice Boltzmann model for time independent flows. *J. Stat. Phys.*, 81:35–48, 1995.
- [106] Yong Shi, T. S. Zhao, and S. Succi. Lattice Boltzmann simulation of dense gas flows in microchannels. *Phys. Rev. E*, 76:016707.1–016707.8, 2007.
- [107] R. S. Maier, R. S. Bernard, and D. W. Grunau. Boundary conditions for the lattice Boltzmann method. *Phys. Fluids*, 8(7):1788–1801, 1996.
- [108] Z. Guo, C. Zheng, and T. S. Zhao. Lattice Boltzmann model for incompressible flows through porous media. *Phys. Rev. E*, 66:036304.1–036304.9, 2002.
- [109] Z. Guo, C. Zheng, and T. S. Zhao. A Lattice BGK Scheme with General Propagation. *J. Sci. Comput*, 16(4):569–585, 2001.
- [110] Y. Shi, T. S. Zhao, and Z. L. Guo. Lattice Boltzmann method for incompressible flows with large pressure gradients. *Phys. Rev. E*, 73:026704.1–026704.11, 2006.
- [111] R. Verberg and A. J. C. Ladd. Simulation of low-Reynold-number flow via a time-independent lattice-Boltzmann method. *Phys. Rev. E*, 60(3):3366–3373, 1999.
- [112] J. Tölke, M. Krafczyk, and E. Rank. A Multigrid-Solver for the Discrete Boltzmann Equation. *J. Stat. Phys.*, 107:573–591, 2002.
- [113] Z. Guo, T. S. Zhao, and Y. Shi. Preconditioned lattice-Boltzmann method for steady flows. *Phys. Rev. E*, 70:066706.1–066706.8, 2004.

- [114] A. M. M. Artoli, L. Abrahamyan, and A. G. Hoekstra. Accuracy versus Performance in Lattice Boltzmann BGK Simulations of Systolic Flows. In M. T. Bubak, G. D. van Albada, P. M. A. Sloot, and J. J. Dongarra, editors, *Computational Science - ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part IV*, Lect. Notes Comput. Sci., pages 548–555. Springer Verlag, 2004.
- [115] R. Zhang, H. Chen, Y. H. Qian, and S. Chen. Effective volumetric lattice Boltzmann scheme. *Phys. Rev. E*, 63(5):056705.1–056705.6, 2001.
- [116] D. d’Humières, I. Ginzburg, P. Lallemand M. Krafczyk, and L.-S. Luo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *J. Comput. Phys.*, 460(1792):437–451, 2002.
- [117] R. Benzi, S. Succi, and M. Vergassola. The lattice Boltzmann equation: theory and applications. *Phys. Rept.*, 222(3):145–197, 2002.
- [118] P. J. Dellar. Bulk and shear viscosities in lattice Boltzmann equations. *Phys. Rev. E*, 64:031203.1–031203.11, 2001.
- [119] F. Nannelli and S. Succi. The lattice Boltzmann equation on irregular lattices. *J. Stat. Phys.*, 68:401–407, 1992.
- [120] I. V. Karlin, S. Succi, and S. Orszag. Lattice Boltzmann Method for Irregular Grids. *Phys. Rev. Lett.*, 82(26):5245–5248, 1999.
- [121] Y. T. Chew and C. Shu and X. D. Niu. A New Differential Lattice Boltzmann Equation and Its Application to Simulate Incompressible Flows on Non-Uniform Grids. *J. Stat. Phys.*, 107:329–342, 2002.
- [122] X. He and G. Doolen. Lattice Boltzmann Method on Curvilinear Coordinates System: Flow around a Circular Cylinder. *J. Comput. Phys.*, 134(2):306–315, 1997.
- [123] R. Mei and W. Shyy. On the finite difference-based lattice Boltzmann method in curvilinear coordinates. *J. Comput. Phys.*, 134(2):426–448, 1998.
- [124] O. Filippova and D. Hänel. Grid refinement for lattice-BGK models. *J. Comput. Phys.*, 147:219–228, 1998.
- [125] C.-L. Lin and L. G. Yong. Lattice Boltzmann method on composite grids. *Phys. Rev. E*, 62(2):2219–2225, 2000.
- [126] B. D. Kandhai, W. Soll, S. Chen, A. G. Hoekstra, and P. M. A. Sloot. Finite-Difference Lattice-BGK Methods on Nested Grids. *Comput. Phys. Commun.*, 129:100–109, 2000.

- [127] D. Yu, R. Mei, and W. Shyy. A multi-block lattice Boltzmann method for viscous fluid flows. *Comput. Phys. Commun.*, 39(2):99–120, 2002.
- [128] O. Filippova, B. Schwade, and D. Hänel. Multiscale Lattice Boltzmann Schemes for Low Mach Number Flows. *Phil. Trans.: Math. Phys. Eng. Sci.*, 360(1792):467–476, 2002.
- [129] M. Rohde, D. Kandhai, J. J. Derksen, and H. E. A. van den Akker. A generic, mass conservative local grid refinement technique for lattice-boltzmann schemes. *Int. J. Num. Meth. Fluids*, 51:439–468, 2006.
- [130] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics A generic, mass conservative local grid refinement technique for lattice-Boltzmann schemes. *J. Comput. Phys.*, 82:64–84, 1989.
- [131] J. J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics*. PhD thesis, Cranfield Institute of Technology, 1991.
- [132] Z. Guo and T. S. Zhao. Explicit finite-difference lattice Boltzmann method for curvilinear coordinated. *Phys. Rev. E*, 67:066709.1–066709.12, 2003.
- [133] H. Chen. Volumetric formulation of the lattice Boltzmann method for fluid dynamics: Basic concept. *Phys. Rev. E*, 58(3):3955–3963, 1998.
- [134] H. Xi, G. Peng, and S.-H. Chou. Finite-volume lattice Boltzmann method. *Phys. Rev. E*, 59(5):6202–6205, 1999.
- [135] G. Peng, C. Duncan H. Xi, and S.-H. Chou. Finite volume scheme for the lattice Boltzmann method on unstructured meshes. *Phys. Rev. E*, 59(4):4675–4682, 1999.
- [136] J. Tölke, M. Krafczyk, M. Shulz, and E. Rank. Finite volume scheme for the lattice Boltzmann method on unstructured meshes. *Phys. Rev. E*, 129(1):91–99, 2000.
- [137] X. Shi, J. Lin, and Z. Yu. Discontinuous Galerkin spectral element lattice Boltzmann method on triangular element. *Int. J. Numer. Methods Fluids*, 42(1):1249–1261, 2000.
- [138] Y. Li, L. J. Eugene J., and P. K. Basu. Least-squares finite-element lattice Boltzmann method. *Int. J. Numer. Methods Fluids*, 69(6):065701.1–065701.4, 2004.
- [139] T. Inamuro. A lattice kinetic scheme for incompressible viscous flows with heat transfer. *Phil. Trans.: Math. Phys. Eng. Sci.*, 360(1792):477–484, 2002.

- [140] M. Junk and S. V. R. Rao. A new discrete velocity method for Navier-Stokes equations. *J. Comput. Phys.*, 155:178, 1999.
- [141] C. Shu, Y. T. Chew, and X. D. Niu. Least-squares-based lattice Boltzmann method: A meshless approach for simulation of flows with complex geometry. *Phil. Trans.: Math. Phys. Eng. Sci.*, 64(4):045701.1–045701.4, 2001.
- [142] X. X. Zhang, A. G. Bengough, J. W. Crawford, , and I. M. Young. A lattice BGK model for advection and anisotropic dispersion equation. *Adv. Water Resour.*, 25:1–8, 2001.
- [143] X. Zhang, L. K. Deeks, A. G. Bengough, J. W. Crawford, and I. M. Young. Determination of soil hydraulic conductivity with the lattice Boltzmann method and soil thin-section technique. *J. Hydrol.*, 306(1–4):59–70, 2005.
- [144] S. Donath, T. Zeiser, G. Hager, J. Habich, and G. Wellein. Optimizing Performance of the Lattice Boltzmann Method for Complex Structures on Cache-based Architectures. In F. Huelsemann, M. Kowarschik, and U. Ruede, editors, *Frontiers in Simulation: Simulation Techniques - 18th Symposium in Erlangen, September 2005 (ASIM)*, SCS Publishing House, Erlangen, pages 728–735, 2005.
- [145] R. Argentini, A. F. Bakker, and C. P. Lowe. Efficiently using memory in lattice Boltzmann simulations. *Fut. Gen. Comput. Syst.*, 20(6):973–980, 2004.
- [146] M. Kowarschik and C. Weiß. An Overview of Cache Optimization Techniques and Cache-Aware Numerical Algorithms. In Ulrich Meyer, Peter Sanders, and Jop F. Sibeyn, editors, *Algorithms for Memory Hierarchies*, volume 2625 of *Lect. Notes Comput. Sci.*, pages 213–232. Springer, 2002.
- [147] T. Pohl, M. Kowarschik, J. Wike, K. Iglberger, and U. Rude. Optimization and Profiling of the Cache Performance of Parallel Lattice Boltzmann Codes in 2D and 3D. *Parallel Processing Letters*, 13(4):549–560, 2003.
- [148] M. Schulz, M. Krafczyk, J. Tölke, and E. Rank. Parallelization strategies and efficiency of CFD computations in complex geometries using Lattice Boltzmann methods on high-performance computers. In M. Breuer, F. Durst, and C. Zenger, editors, *in High-Performance Scientific and Engineering Computing, Proceedings of the 3rd International FORTWIHR Conference on HPSEC, Erlangen, 2001*, page 115122. Springer-Verlag, 2002.
- [149] N. S. Martys and J. G. Hagedorn. Multiscale Modeling of Fluid Transport in Heterogeneous Materials Using Discrete Boltzmann Methods. *Mater. Struct.*, 35:650–659, 2006.

- [150] G. Wellein, T. Zeiser, G. Hager, and S. Donath. On the Single Processor Performance of Simple Lattice Boltzmann Kernels. *Comput. Fluids*, 35(8-9), 2006.
- [151] K. Iglberger. *Cache Optimizations for the Lattice Boltzmann Method in 3D*. PhD thesis, Bachelor thesis, Institut für Informatik, University of Erlangen-Nuremberg, Nuremberg, Germany, 2003.
- [152] A. Velivelli and K. M. Bryden. A cache-efficient implementation of the lattice Boltzmann method for the two-dimensional diffusion equation. *Concurrency and Computation: Practice and Experience*, 16(14):1415–1432, 2004.
- [153] S. Donath. *On Optimized Implementations of the Lattice Boltzmann Method on Contemporary High Performance Architectures*. PhD thesis, Bachelor thesis, University of Erlangen-Nuremberg, Erlangen, Germany, 2004.
- [154] M. Kowarshik. *Data Locality Optimizations for Iterative Numerical Algorithms and Cellular Automata on Hierarchical Memory Architectures*. PhD thesis, PhD thesis, University of Erlangen-Nuremberg, Erlangen, Germany, 2004.
- [155] S. Hausmann. *Optimization and Performance Analysis of the Lattice Boltzmann Method on x86-64 based Architecture*. PhD thesis, Bachelor thesis, University of Erlangen-Nuremberg, Erlangen, Germany, 2004.
- [156] G. Hager, T. Zeiser, J. Treibig, and G. Wellein. *Computational Science and High Performance Computing II, Optimizing performance on modern HPC systems: learning from simple kernel benchmarks*, pages 273–287. Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Springer Berlin / Heidelberg, 2006.
- [157] J.-C. Desplat, I. Pagonabarraga, and P. Bladon. Ludwig: A parallel Lattice-Boltzmann code for complex fluids. *Comput. Phys. Commun.*, 134(3):273–290, 2001.
- [158] G. Punzo, F. Massaioli, and S. Succi. Lattice-Boltzmann hydrodynamics on parallel systems. *Comput. Phys.*, 8(6):705–711, 1994.
- [159] G. Amati, S. Succi, and R. Piva. Massively Parallel Lattice-Boltzmann Simulation of Turbulent Channel Flow. *Int. J. Mod. Phys. C*, 8(4):869–877, 1997.
- [160] G. Karypis and V. Kumar. Parallel Multilevel k -way Partitioning Scheme for Irregular Graphs. In *CD-ROM Proceedings of SC '96*, Pittsburgh, PA, 1996. IEEE.
- [161] G. Karypis and V. Kumar. Multilevel k -way Partitioning Scheme for Irregular Graphs. *J. Parallel Distrib. Comput.*, 48(1):96–129, 1998.

- [162] G. Karypis, K. Schloegel, and V. Kumar. *ParMeTis: Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 2.0*. University of Minnesota, Dept. of Computer Science, 1999.
- [163] A. Dupuis and B. Chopard. Lattice Gas: An Efficient and Reusable Parallel Library Based on a Graph Partitioning Technique. *Lect. Notes Comput. Sci.*, 1593:319–328, 1999.
- [164] L. Axner, J. M. Bernsdorf, T. Zeiser, P. Lammers, J. Linxweiler, and A. G. Hoekstra. Performance evaluation of a parallel sparse lattice Boltzmann solver. *J. Comput. Phys.*, 227(10):4895–4911, 2008.
- [165] C. Pan, J. F. Prins, and C. T. Miller. A high-performance lattice Boltzmann implementation to model flow in porous media. *Comput. Phys. Commun.*, 158(2):89–105, 2004.
- [166] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing*. Benjamin/Cummings, 1993.
- [167] H. Sagan. *Space-Filling Curves*. Springer-Verlag, New York, NY, 1994.
- [168] M. J. Aftosmis, M. J. Berger, and S. M. Murman. Applications of Space-Filling Curves to Cartesian Methods for CFD. Technical Report AIAA 2004-1232, American Institute of Aeronautics and Astronautics, 1801 Alexander Bell Drive, Suite 500, Reston, VA 20191, USA, 2004.
- [169] M. D. Mazzeo and P. V. Coveney. HemeLB: A high performance parallel lattice-Boltzmann code for large scale fluid flow in complex geometries. *Comput. Phys. Commun.*, 178(12):894–914, 2008.
- [170] M. Junk and Z. Yang. Asymptotic Analysis of Lattice Boltzmann Boundary Conditions. *J. Stat. Phys.*, 121:3–35, 2005.
- [171] Z. Yang. Pressure condition for lattice Boltzmann methods on domains with curved boundaries. *Comput. Math. Appl.*, 2009.
- [172] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows. *Comput. Fluids*, 35:888–897, 2006.
- [173] N. Thürey and U. Rüde. Optimized free surface fluids on adaptive grids with the lattice Boltzmann method. In *MICCAI '99: Proceedings of the Second International Conference on Medical Image Computing and Computer-Assisted Intervention, SIGGRAPH Poster*, volume 35. ACM, NY, USA, 2005.

- [174] J. Tölke, S. Freudinger, and M. Krafczyk. An adaptive scheme using hierarchical grids for lattice Boltzmann multi-phase flow simulations. *Comput. Fluids*, 35:820–830, 2006.
- [175] D. Kandhai and A. Koponen and A. G. Hoekstra and M. Kataja and J. Timonen and P. M. A. Sloot. Lattice-Boltzmann hydrodynamics on parallel systems. *Comput. Phys. Commun.*, 111:1167–1176, 1998.
- [176] A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.
- [177] T. Goehring and Y. Saad. Heuristic algorithms for automatic graph partitioning. Technical Report UMSI 94-29, University of Minnesota Supercomputer Institute, Minneapolis, MN 55415, 1994.
- [178] P. Ciarlet, Jr. and F. Lamour. On the validity of a front-oriented approach to partitioning large sparse graphs with a connectivity constraint. *Numerical Algorithms*, 12(1–2):193–214, 1996.
- [179] S. Dong, G. E. Karniadakis, and N. T. Karonis. Cross-Site Computations on the TeraGrid. *Comput. Sci. and Eng.*, 7(5):14–23, 2005.
- [180] B. M. Boghosian, P. V. Coveney, S. Dong, L. Finn, S. Jha, G. E. Karniadakis, and N. T. Karonis. NEKTAR, SPICE and Vortronics: using federated grids for large scale scientific applications. *Cluster Comput.*, 10(3):351–364, 2007.
- [181] S. Jha, P. V. Coveney, and M. J. Harvey. SPICE: Simulated Pore Interactive Computing Environment. In *SC' 05*, page 70. IEEE Computer Society, 2005.
- [182] B. M. Boghosian, L. I. Finn, and P. V. Coveney. Moving the data to the computation: multi-site distributed parallel computation. <http://www.realitygrid.org/publications.shtml>.
- [183] N. Karonis, B. Toonen, and I. Foster. A Grid-Enabled Implementation of the Message Passing Interface. *J. Parallel Distrib. Comput.*, 63(5):551–563, 2003.
- [184] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. R. Toonen. Distributed MPI Cross-site Run Performance Using MPIg. In M. Parashar, K. Schwan, J. B. Weissman, and D. Laforenza, editors, *Proceedings of High Performance Distributed Computing (HPDC'08), June 23-27*, pages 229–230. ACM, 2008.
- [185] <http://www.globus.org/>.
- [186] <http://www.hpcx.ac.uk/>.
- [187] <http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/Intel64Cluster/>.

- [188] <http://www.loni.org/systems/>.
- [189] X. Wu, V. E. Taylor, S. Garrick, D. Yu, and J. Richard. Performance Analysis, Modeling and Prediction of a Parallel Multiblock Lattice Boltzmann Application Using Prophecy System. In *CLUSTER*. IEEE, 2006.
- [190] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach, 2nd Edition*. M. Kaufmann, 1996.
- [191] G. Knittel. The ULTRAVIS system. In *IEEE/ACM SIGGRAPH Volume visualization and graphics symposium 2000*, pages 71–78, 2000.
- [192] B. Mora, J.-P. Jessel, and R. Caubet. A New Object-Order Ray-Casting Algorithm. In *IEEE Visualization*, 2002.
- [193] S. Grimm, S. Bruckner, A. Kanitsar, and Eduard Gröller. Memory Efficient Acceleration Structures and Techniques for CPU-Based Volume Raycasting of Large Data. In *VolVis*, pages 1–8. IEEE Computer Society, 2004.
- [194] M. Sramek and A. Kaufman. Fast Ray-Tracing of Rectilinear Volume Data Using Distance Transforms. In H. Hagen, editor, *IEEE Trans. Vis. Comput. Graph.*, volume 6 (3), pages 236–252. IEEE Computer Society, 2000.
- [195] M. Wan, A. Sadiq, and A. Kaufman. Fast and Reliable Space Leaping for Interactive Volume Rendering. In R. Moorhead, M. Gross, and K. I. Joy, editors, *Proceedings of the 13th IEEE Visualization 2002 Conference (VIS-02)*, pages 195–202, Piscataway, NJ, 2002. IEEE Computer Society.
- [196] I. Wald, H. Friedrich, G. Marmitt, P. Slusallek, and H.-P. Seidel. Faster Isosurface Ray Tracing Using Implicit KD-Trees. *IEEE Trans. Vis. Comput. Graph.*, 11(5):562–572, 2005.
- [197] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *IEEE Visualization '98*, pages 233–238, Washington - Brussels - Tokyo, 1998. IEEE.
- [198] M. Gross, C. Lojewski, M. Bertram, and H. Hagen. Fast implicit kd-trees: accelerated isosurface ray tracing and maximum intensity projection for large scalar fields. In *Proceedings of Computer Graphics and Imaging (CGIM) 2007*, pages 67–74, 2007.
- [199] B. Cabral and N. Cam and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *In VVS 94: Proceedings of the 1994 Symposium on Volume Visualization (1994)*, ACM Press, pages 91–98, 1995.

- [200] K. Engel, M. Kraus, and T. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *In Eurographics / SIGGRAPH Workshop on Graphics Hardware 01 (2001), Annual Conference Series, Addison-Wesley Publishing Company, Inc.*, pages 9–16, 2001.
- [201] P. Ljung. *Efficient Methods for Direct Volume Rendering of Large Data Sets*. PhD thesis, Linköping University, Sweden, 2006.
- [202] M. Levoy. Efficient Ray Tracing of Volume Data. *ACM Transactions on Graphics*, 9(3):245–61, 1990.
- [203] Y. Livnat and C. D. Hansen. View dependent isosurface extraction. In *IEEE Visualization*, pages 175–180, 1998.
- [204] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111, 1999.
- [205] A. Knoll, C. D. Hansen, and I. Wald. Coherent Multiresolution Isosurface Ray Tracing. Technical Report UUSCI-2007-001, SCI Institute, University of Utah, 2007. (submitted for publication).
- [206] H. Friedrich, I. Wald, J. Günther, G. Marmitt, and P. Slusallek. Interactive Iso-Surface Ray Tracing of Massive Volumetric Data Sets. In J. M. Favre, L. P. Santos, and D. Reiners, editors, *Eurographics Symposium on Parallel Graphics and Visualization*, pages 109–116, Lugano, Switzerland, 2007. Eurographics Association.
- [207] M. Matsui, F. Ino, and K. Hagihara. Parallel Volume Rendering with Early Ray Termination for Visualizing Large-Scale Datasets. In J. Cao, L. T. Yang, M. Guo, and F. C.-M. Lau, editors, *Parallel and Distributed Processing and Applications, Second International Symposium, ISPA 2004, Hong Kong, China, December 13-15, 2004, Proceedings*, volume 3358 of *Lect. Notes Comput. Sci.*, pages 245–256. Springer, 2004.
- [208] Y. Wu, V. Bhatia, H. C. Lauer, and L. Seiler. Shear-image order ray casting volume rendering. In *In SI3D 03: Proceedings of the 2003 symposium on Interactive 3D graphics (2003)*, pages 152–162, 2003.
- [209] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH*, pages 451–458. ACM, 1994.
- [210] G. Marmitt, R. Brauchle, H. Friedrich, and P. Slusalle. Accelerate and Extended Building of Implicit kd-Trees for Volume Ray Tracing. In L. Kobbelt, T. Kulen,

- and R. Westermann, editors, *Proceedings of 11th International Fall Workshop - Vision, Modeling, and Visualization (VMV) 2006*, pages 317–324, Aachen, Germany, 2006. Akademische Verlagsgesellschaft Aka.
- [211] H.-W. Shen, L.-J. Chiang, and K.-L. Ma. A Fast Volume Rendering Algorithm for Time-Varying Fields Using a Time-Space Partitioning (TSP) Tree. In D. Ebert, M. Gross, and B. Hamann, editors, *IEEE Visualization '99*, pages 371–378, San Francisco, 1999. IEEE.
- [212] H. Yu, C. Wang, and K.-L. Ma. Parallel Hierarchical Visualization of Large 3D Time-Varying Vector Fields. In *In Proceedings of ACM/IEEE Supercomputing 2007 Conference (SC '07)*, 2007.
- [213] A. Knoll, I. Wald, S. G. Parker, and C. D. Hansen. Interactive Isosurface Ray Tracing of Large Octree Volumes. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pages 115–124, 2006.
- [214] P. M. S. and C. D. Hansen. Accelerated Isosurface Extraction in Time-Varying Fields. *IEEE Trans. Vis. Comput. Graph.*, 6(2):98–107, 2000.
- [215] E. Reinhard, C. Hansen, and S. Parker. Interactive Ray Tracing of Time Varying Data. In S. N. Spencer, editor, *Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization (EGPGV-02)*, pages 77–82, New York, 2002. ACM Press.
- [216] M. Strengert, M. Magallón, D. Weiskopf, S. Guthe, and T. Ertl. Hierarchical Visualization and Compression of Large Volume Datasets Using GPU Clusters. In B. Raffin, D. Bartz, and H.-W. Shen, editors, *Eurographics Symposium on Parallel Graphics and Visualization*, pages 41–48, Grenoble, France, 2004. Eurographics Association.
- [217] A. Stompel, K.-L. Ma, E. B. Lum, J. P. Ahrens, and J. Patchett. SLIC: Scheduled Linear Image Compositing for Parallel Volume Rendering. In A. H. J. Koning, R. Machiraju, and C. T. Silva, editors, *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003 (PVG'03)*, pages 33–40, Seattle, WA, USA, 2003. IEEE Computer Society (Los Alamitos, CA).
- [218] E. B. Lum, K.-L. Ma, and J. Clyne. Texture Hardware Assisted Rendering of Time-Varying Volume Data. In T. Ertl, K. Joy, and A. Varshney, editors, *Proceedings Visualization 2001*, pages 263–270. IEEE Computer Society Technical Committee on Visualization and Graphics Executive Committee, 2001.

- [219] E. B. Lum, K.-L. Ma, and J. Clyne. A Hardware-Assisted Scalable Solution for Interactive Volume Rendering of Time-Varying Data. *IEEE Trans. Vis. Comput. Graph.*, 8(3):286–301, 2002.
- [220] R. Van Liere, J. D. Mulder, and J. J. Van Wijk. Computational Steering. *Lect. Notes Comput. Sci.*, 1067:696, 1996.
- [221] S. M. Pickles, R. Haines, R. L. Pinning, and A. R. Porter. A Practical Toolkit for Computational Steering. *Phil. Trans. Roy. Soc. A*, 363(1833):1843–1853, 2005.
- [222] H. Yu, K.-L. Ma, and J. Welling. A Parallel Visualization Pipeline for Terascale Earthquake Simulations. In *SC' 04 Conference CD*, Pittsburgh, PA, 2004. IEEE/ACM SIGARCH.
- [223] R. G. Bellemann and R. Shulakov. High Performance Distributed Simulation for Interactive Simulated Vascular Reconstruction. *Lect. Notes Comput. Sci.*, 2331:265, 2002.
- [224] J. A. Insley, M. E. Papka, S. Dong, G. E. Karniadakis, and N. T. Karonis. Runtime Visualization of the Human Arterial Tree. *IEEE Trans. Vis. Comput. Graph.*, 13(4):810–821, 2007.
- [225] J. S. Rowlan, G. E. Lent, N. Gokhale, and S. Bradshaw. A Distributed, Parallel, Interactive Volume Rendering Package. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of the Conference on Visualization*, pages 21–30, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.
- [226] T. Takei, J. Bernsdorf, N. Masuda, and H. Takahara. Lattice Boltzmann Simulation and its Concurrent Visualization on the SX-6 Supercomputer. In *High Performance Computing and Grid in Asia Pacific Region, Seventh International Conference on (HPCAAsia'04)*, pages 212–219. IEEE, 2004.
- [227] K.-L. Ma. Runtime Volume Visualization for Parallel CFD. Technical Report 95-74, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, USA, 1995.
- [228] P. S. Lomdahl. Multibillion-atom Molecular Dynamics Simulations on BlueGene/L. *Bulletin Amer. Phys. Soc. 2005. Meeting Abstracts*, 50(5):5003, 2005.
- [229] T. Tu, H. Yu, L. R.-G., J. Bielik, O. Ghattas, K.-L. Ma, and D. R. O'Hallaron. From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing. In *SC' 06 Conference CD*, Tampa, FL, 2006. IEEE/ACM SIGARCH.

- [230] Message Passing Interface Forum. MPI: A Message Passing Interface. In *Proc. SC '93*, pages 878–883, Portland, OR, USA, 1993. IEEE Computer Society.
- [231] W. M. Hsu. Segmented Ray Casting for Data Parallel Volume Rendering. In T. Crockett, C. Hansen, and S. Whitman, editors, *ACM SIGGRAPH Symposium on Parallel Rendering*, pages 6–14. ACM, 1993.
- [232] A. Neubauer, L. Mroz, H. Hauser, and R. Wegenkittl. Cell-based First-hit Ray Casting. In D. Ebert, P. Brunet, and I. Navazo, editors, *Proceedings of the symposium on Data visualisation*, pages 077–086, Barcelona, Spain, 2002. Eurographics Association.
- [233] J. Amanatides and A. Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics '87*, pages 3–10. Elsevier Science Publishers, Amsterdam, North-Holland, 1987.
- [234] J. MacLaren, M. Mc Keown, and S. Pickles. Co-Allocation, Fault Tolerance and Grid Computing. In *Proceedings of the UK e-Science All Hands Meeting 2006*, pages 155–162, 2006.
- [235] *An interactive grid environment for non-invasive vascular reconstruction*, 2004.
- [236] L. Abrahamyan, J. A. Schaap, A. G. Hoekstra, D. P. Shamonin, F. M. A. Box, Rob J. van der Geest, Johan H. C. Reiber, and P. M. A. Sloot. A Problem Solving Environment for Image-Based Computational Hemodynamics. In Vaidy S. Sunderam, G. Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *Computational Science - ICCS 2005, 5th International Conference, Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part I*, volume 3514 of *Lect. Notes Comput. Sci.*, pages 287–294. Springer, 2005.
- [237] A. Tirado-Ramos, G. Tsouloupas, M. D. Dikaiakos, and P. M. A. Sloot. Grid Resource Selection by Application Benchmarking for Computational Haemodynamics Applications. In Vaidy S. Sunderam, G. Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *Computational Science - ICCS 2005, 5th International Conference, Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part I*, volume 3514 of *Lect. Notes Comput. Sci.*, pages 534–543. Springer, 2005.
- [238] K. Rycerz, M. Bubak, M. Malawski, and P. M. A. Sloot. Grid Support for HLA-Based Collaborative Environment for Vascular Reconstruction. In *eScience*, page 48. IEEE Computer Society, 2006.
- [239] <http://www.eu-crossgrid.org/>.
- [240] <http://www.grid-support.ac.uk/>.
- [241] <http://www.teragrid.org/>.

- [242] P. V. Coveney, R. S. Saksena, S. J. Zasada, M. McKeown, and S Pickles. The Application Hosting Environment: Lightweight Middleware for Grid-based Computational Science. *Comput. Phys. Commun.*, 176(6):406–418, 2007.
- [243] S. J. Zasada, R. Saksena, P. V. Coveney, M. Mc Keown, and S. Pickles. Facilitating User Access to the Grid: A Lightweight Application Hosting Environment for Grid Enabled Computational Science. In *Second IEEE International Conference on e-Science and Grid Computing, 2006*, pages 50–58, 2006.
- [244] S. J. Zasada and P. V. Coveney. From campus resources to federated international grids: bridging the gap with the application hosting environment. In *GCE '09: Proceedings of the 5th Grid Computing Environments Workshop*, pages 1–10, 2009.
- [245] GENIUS project wiki. <http://wiki.realitygrid.org/wiki/GENIUS>.
- [246] S. Manos, S. Zasada, and P. V. Coveney. Life or Death Decision-making: The Medical Case for Large-scale, On-demand Grid Computing. *CTWatch Quarterly Journal*, 4(2):35–45, 2008.
- [247] P. Beckmann. Urgent Computing: Exploring Supercomputing’s New Role. *CT-Watch Quarterly Journal*, 4(2), 2008.
- [248] K. Yoshimoto, P. Kovatch, and P. Andrews. Co-scheduling with User-Settable Reservations. In D. G. Feitelson, E. Frachtenberg, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 146–156. Springer Verlag, 2005. Lect. Notes Comput. Sci. vol. 3834.
- [249] D. Marcusiu, M. Margo, K. Yoshimoto, and P. Kovatch. Automatic Co-Scheduling on the TeraGrid. In *Proceedings of the 2006 TeraGrid Conference*, 2006.
- [250] SDSC IA-64 cluster. <http://www.sdsc.edu/us/resources/ia64>.
- [251] Siemens Artis Zee multipurpose imaging system. <http://www.siemens.com/artis-zee>.
- [252] O. Wirjadi. Survey of 3D Image Segmentation Methods. Technical Report 123, Models and Algorithms in Image Processing Fraunhofer ITWM, Kaiserslautern, 2007.
- [253] Q. Zou and X. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *J. Phys. Fluids*, 9(6):1591–1598, 1997.
- [254] C. Fan and B.-T. Chao. Unsteady, laminar, incompressible flow through rectangular ducts. *Zeitschrift für Angewandte Mathematik und Physik*, 16(3):351–360, 1965.

- [255] B. D. Kandhai, A. Koponen, A. G. Hoekstra, M. Kataja, J. Timonen, and P. M. A. Sloot. Implementation Aspects of 3D Lattice-BGK: Boundaries, Accuracy, and a New Fast Relaxation Method. *J. Comput. Phys.*, 150:482–501, 1999.
- [256] David A. Lane. Visualizing Time-Varying Phenomena In Numerical Simulations Of Unsteady Flows. Technical Report NAS-96-001, NASA Ames Research Center, 1996.
- [257] <http://www.itk.org/>.
- [258] <http://www.ansys.com/Products/cfx.asp>.
- [259] K. Ryan and G. Sheard. Wall shear stress and flow stability in a model fusiform aneurysm. In G. Mercer and A. J. Roberts, editors, *Proceedings of the Computational Techniques and Applications Conference, Australian National University, Canberra, Australia*, volume 50. Cambridge University Press, 2009.
- [260] J. A. Moore, D. A. Steinman, and C. R. Ethier. Computational blood flow modelling: Errors associated with reconstructing finite element models from magnetic resonance images. *J. Biomech.*, 31(2):179–184, 1998.