A Parallel Lattice Boltzmann Model of a Carotid Artery

J. Boyd, S. J. Ryan and J. M. Buick

Physics, School of Science & Technology, University of New England, Armidale, NSW, 2351, Australia

Abstract. A parallel implementation of the lattice Boltzmann model is considered for a three dimensional model of the carotid artery. The computational method and its parallel implementation are described. The performance of the parallel implementation on a Beowulf cluster is presented, as are preliminary hemodynamic results.

Keywords: Lattice Boltzmann; parallel processing; arterial simulation
PACS: 47.11.-j, 87.19.U-

INTRODUCTION

The lattice Boltzmann model (LBM) [1] is a relatively new approach to fluid simulation which has been applied to a wide range of problems in fluid mechanics. Recently it has been applied successfully to study arterial flow in the carotid artery [2–5]. A number of aspects were investigated such as the importance of the non Newtonian nature of blood, the wall stresses, and the changing hemodynamic properties during stenosis growth. This has provided an insight into the complex processes involved, however, one limitation of this work is that it was performed using a two dimensional model. This paper considers the development of a three dimensional model and its parallel implementation. LBM simulations of arterial blood flow have also been considered by a number of other authors [6–15].

THE LATTICE BOLTZMANN METHOD

The LBM [1] is performed on a regular grid in either two or three dimensions. Each grid point is connected to n-1 neighbors and the model is labeled DdQn, where d is the number of dimensions. At each grid point the fluid is described in terms of n distribution functions f_i, for i = 0, 1, ..., n, each associated with a vector $\mathbf{e_i}$ connecting the grid point to one of its n-1 neighbors; or, for i = 0, $\mathbf{e_0}$ is the null vector. The fluid evolves according to the LBM equation

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

where τ is the relaxation time and the equilibrium distribution function, f_i^{eq} is determined as a function of the local density $\rho = \sum_i f_i$ and velocity $\mathbf{u} = \sum_{\mathbf{i}} \mathbf{f_i e_i}/\rho$. The left hand side of Eq. (1) represent streaming of the distribution functions and the right hand side is the collision operator which mimics particle collisions. Here the D2Q9 and D3Q15 lattices are used, where $\mathbf{e_i}$ is a unit vector along the directions of the positive and negative axis for i = 1, ..., 2d and a diagonal vector of length $\sqrt{d}$ for $i = 2d+1, ..., n-1$. For both these grids [16]

$$f_i^{eq} = w_i\rho\left(1 + 3\mathbf{e}_i\cdot\mathbf{u} + \frac{9}{2}(\mathbf{e}_i\cdot\mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2\right), \tag{2}$$

where w_i are weight functions given by 4/9 (2/9), 1/9 (1/9) and 1/36 (1/72) for the null, axial and diagonal directions respectively for the D2Q9 (D3Q15) model. The kinematic viscosity of the fluid is $\nu = (2\tau - 1)/6$.

COMPUTATIONAL REQUIREMENTS, HARDWARE AND IMPLEMENTATION

The earlier arterial work presented by the authors was performed using the D2Q9 model. To perform a simulation (with the same resolution) in three dimensions additional computational resources are required due to:
1. The additional number of grid points. The additional number of grid points depends on the geometry of the artery. The section of artery simulated was considerably longer than the diameter of the artery which varied along the length.

Further, the artery section included a bifurcation (the geometry of the artery can be seen in the results presented in Fig. 3). The geometry of the artery was described in three dimensional space and was not orientated along any of the axis. Thus the volume of a cubic box (with edges parallel to the axis) was considerably larger than the actual volume of the artery. To overcome this the code was written in such a way that only grid points inside the artery were considered in the simulation (except during initialization). This was achieved by labeling each grid point in the simulation and looping through a one dimensional list of grid points. All points in the cross section $z = z_{min}$ were labeled first, followed by $z = z_{min} + 1$, ..., $z = z_{max}$, where z_{min} and z_{max} are the minimum and maximum values of z and the z axis is closest to the axis of the artery, see Fig. 1.

2. The increased number of link directions. Using the D3Q15 model the number of distribution function which must be calculated is increased by a factor slightly less than two.

3. The extra calculations required to determine the three components of the velocity and the terms containing the velocity in the expression for the equilibrium distribution function, Eq. (2).

A parallel implementation of the LBM was developed to enable the simulation of large three dimensional geometries. A number of properties make the LBM highly suitable for parallel implementation [17–20], these include the local nature of the collision operator and the fact that nearest neighbour nodes only interact during the streaming step of the algorithm. In parallel applications, computational tasks are subdivided and distributed to a number of processors. In this way, each processor is required to do less calculation, therefore decreasing the total simulation time. However, communication between processors is computationally expensive [21, 22]; thus an algorithm that minimizes communications is required.

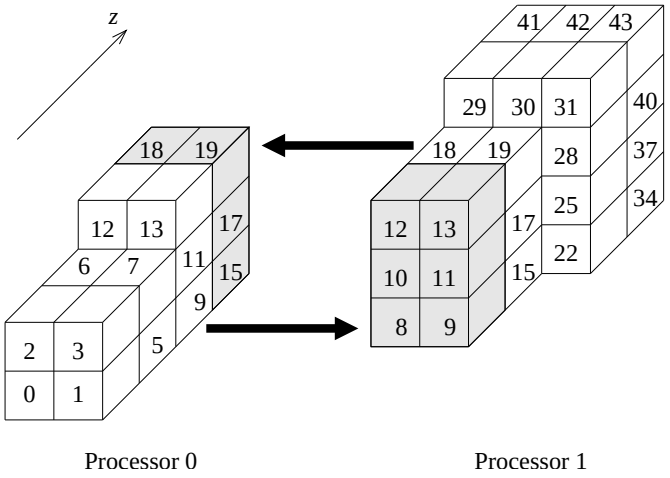

FIGURE 1. Lattice division between two processors.

Figure 1 shows a forty four node ‘artery’ divided between two processors. The artery has a non-regular shape and each node is labeled sequentially. If the total domain contains Z layers (6 in Fig. 1) in the z direction then it is divided such that each of the n processors contains Z/n layers (3 in Fig. 1). This is the calculation domain represented by the unshaded region in Fig. 1, where processor 0 contains nodes 0 - 13 and processor 1 contains nodes 14 - 43. Note that the load is not distributed equally over each processor; this will be discussed in section below. Further, at the interface between each processor a ‘halo’ region is also considered, represented by the shaded region in Fig. 1. These sites contains values which are required during the collision step, but not calculated by the processor. These are communicated from the adjacent processor as indicated by the arrows in Fig. 1. For a long, thin artery section, this approach minimizes the communication requirements.

The parallel simulations were run on the University of New England Beowulf cluster which consists of three species of processors for a total of 24 processors operating on a 100MB fast Ethernet private SAN (Systems Area Network). The three species of processors consisted of the hardware show in Table 1. The parallel LBM code was implemented in C using LAM MPI-2. All simulations run on 1-8 processors used the u-nodes. Simulations on 16 processors used the u- and b-nodes, but timings are given relative to the u-nodes.

TABLE 1. Composition of the UNE Beowulf cluster.

Species Designation	Number of Processors	Type	CPU Speed (MHz)	HDD (GB)	RAM (GB)
o	8	Athlon	1150	34	1.5
u	8	Athlon 64	2000	34	1.0
b	8	Pentium 4	2666	35	0.5

RESULTS AND DISCUSSION

The parallel implementation of the LBM code was verified by simulating Poiseuille flow, enabling a comparison between the parallel simulations, sequential simulations and theory. Two and three dimensional simulations of the carotid artery were also performed and agreement was found between the parallel and sequential codes.

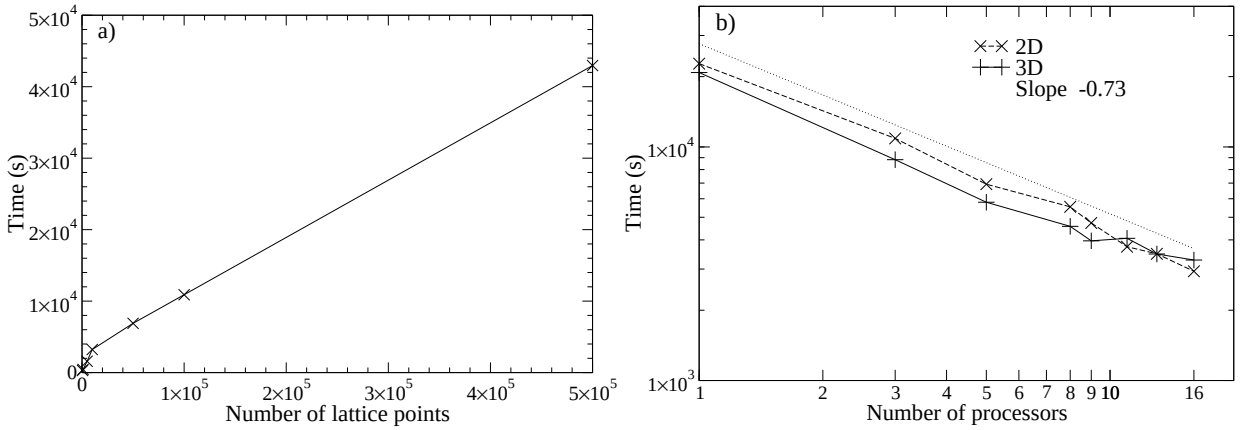

FIGURE 2. Parallel timings excluding initialization. a) Increase in simulation time with increased number of grid points for Poiseuille fbw on 8 processors. b) Decrease in simulation time with number of processors for a fi xed grid size.

Figure 2 shows the characteristics of the parallel system. In Fig. 2 a) the scale-up is demonstrated for 8 processors. The total time is shown for a simulation of 10^6 time steps for a varying number of grid points. The scale-up is approximately linear for more than approximately 10^4 grid points (around 10^3 grid points per processor). This corresponds to the minimum number of grid points per processor required for optimal performance. The speed up for a fixed domain size, significantly larger than 10^3 grid points per processor, is indicated in Fig. 2 b) for a two dimensional (4.5×10^{10} site updates) and a three dimensional (1.4×10^{10} site updates) artery model. Figure 2 b) indicates a speed up of around 0.73. A typical velocity output on a section of the carotid artery is shown in Fig. 3. A detailed analysis of the results of the three dimensional simulations will be presented elsewhere. A speed-up of approximately 0.73 enables large three dimensional simulations to be performed in a realistic time. To ensure the correct physiological and hemodynamic properties and to provide a detailed resolution, a typical simulation contains around 1.5 million grid points. At this resolution a pulse period of around 0.25 million time steps takes approximately 20 hours on the 16 processors and 4-5 periods are required for the simulation to converge.

As notes in Fig. 1, the load is not exactly balanced due to the non-regular shape of the artery. In practice this is not as significant a problems as the example in Fig. 1. Addressing this may produce a slight increase in performance.

SUMMARY

A parallel implementation of a LBM for the carotid artery has been considered. Performance details and preliminary results have been presented.

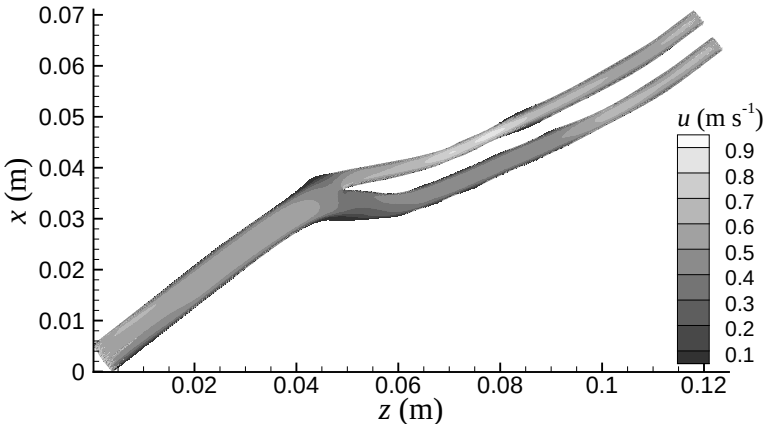

FIGURE 3. Velocity profi le in a section of the carotid artery.

ACKNOWLEDGMENTS

The authors would like to acknowledge the technical assistance of N. Gaywood and A. Saint. J. Boyd would like to acknowledge an APA.

REFERENCES

1. S. Chen, and G. D. Doolen, *Annual Review of Fluid Mechanics* **30**, 329–364 (1998).
2. J. Boyd, J. M. Buick, J. A. Cosgrove, and P. Stansell, *Australasian Physical & Engineering Sciences in Medicine* **27**, 207–212 (2004).
3. J. Boyd, J. Buick, J. A. Cosgrove, and P. Stansell, *Physics in Medicine and Biology* **50**, 4783–4796 (2005).
4. J. Boyd, and J. M. Buick, *Physics in Medicine and Biology* **52**, 6215–6228 (2007).
5. J. Boyd, J. M. Buick, and S. Green, *Physics of Fluids* **19**, 093103 (2007).
6. M. Krafczyk, J. Tolke, E. Rank, and M. Schulz, *Computers & Structures* **79**, 2031–2037 (2001).
7. C. Migliorini, Y. Qian, H. Chen, E. B. Brown, R. K. Jain, and L. L. Munn, *Biophys. J.* **83**, 1834–1841 (2002).
8. H. Fang, Z. Wang, Z. Lin, and M. Liu, *Phys. Rev. E* **65**, 051925 (2002).
9. M. Tamagawa, and S. Matsuo, *JSME International Journal Series C* **47**, 1027–1034 (2004).
10. I. H. M. M. Dupin, and C. M. Care, *Journal of Physics A* **36**, 8517–8534 (2003).
11. A. M. Artoli, D. Kandhai, H. C. J. Hoefsloot, A. G. Hoekstra, and P. M. A. Sloot, *Future Generation Computer Systems* **20**, 909–916 (2004).
12. H. Li, H. Fang, Z. Lin, S. Xu, and S. Chen, *Physical Review E* **69**, 031919 (2004).
13. A. Artoli, A. Hoekstra, and P. Sloot, *Journal of Biomechanics* **39**, 873–884 (2006).
14. J. Bernsdorf, S. E. Harrison, S. M. Smith, P. V. Lawford, and D. R. Hose, *International Journal of Bioinformatics Research and Applications* **2**, 371–380 (2006).
15. O. Pelliccioni, M. Cerrolaza, and M. Herrera, *Math. Comput. Simul.* **75**, 1–14 (2007), ISSN 0378-4754.
16. D. D. Y. H. Qian, and P. Lallemand, *Europhysics Letters (EPL)* **17**, 479–484 (1992).
17. T. Pohl, M. Kowarschik, J. Wilke, K. Iglberger, and U. Rüde, *Parallel Processing Letters* **13**, 549–560 (2003).
18. M. J. Pattison, and S. Banerjee, "Numerical simulation of fluids using the latticeBoltzmann scheme," in *The 6th International Conference on Nuclear Thermal Hydraulics, Operations and Safety (NUTHOS-6)*, Nara, Japan, 2004, p. N6P241.
19. B. He, W.-B. Feng, W. Zhang, and Y.-M. Cheng, "Parallel simulation of compressible fluid dynamics usingn lattice Boltzmann method," in *The first International Symposium on Optimization and System Biology (ISM'07)*, Beijing, China, 2007, pp. 451–458.
20. C. Körner, T. Pohl, U. Rüde, N. Thürey, and T. Zeiser, "Parallel lattice Boltzmann Methods for CFD," in *Numerical Solution of Partial Differential Equations on Parallel Computers*, Springer, 2006, pp. 439–465.
21. A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing: Second Edition*, Pearson/Addison Wesley, 2003.
22. W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, The MIT Press, 1994.