Large Scale Simulations of Single and Multi-Component Flow in Porous Media

Nicos S. Martys[a], John G. Hagedorn[b], Delphine Goujon[c], and Judith E. Devaney[b]

[a]National Institute of Standards and Technology
100 Bureau Drive, Stop 8621
Gaithersburg, MD 20899-8621, USA

[b]National Institute of Standards and Technology
100 Bureau Drive, Stop 8951
Gaithersburg, MD 20899-8951, USA

[c]Télécomm INT
9 rue Charles Fourier
91011 Evry Cedex, France

ABSTRACT

We examine the utility of the lattice Boltzmann method for modeling fluid flow in large microstructures. First, results of permeability calculations are compared to predicted values for several idealized geometries. Large scale simulations of fluid flow through digitized images of Fontainebleau sandstone, generated by X-ray microtomography, were then carried out. Reasonably good agreement was found when compared to experimentally determined values of permeability for similar rocks. We also calculate relative permeability curves as a function of fluid saturation and driving force. The Onsager relation, which equates off-diagonal components of the permeability tensor for two phase flow, is shown not to hold for intermediate to low nonwetting saturation, since the response of the fluid flow to an applied body force was nonlinear. Values of permeability from three phase flows are compared to corresponding two phase values. Performance on several computing platforms is given.

Keywords: lattice Boltzmann, microtomography, parallel computing, permeability, porous media

1. INTRODUCTION

The lattice Boltzmann (LB) method has evolved into a powerful computational method for the modeling of fluid flow in complex geometries like porous media. It naturally accommodates a variety of boundary conditions such as the pressure drop across the interface between two fluids and wetting effects at a fluid-solid interface. Since the LB method can be derived from the Boltzmann equation, its physical underpinnings can be understood from a fundamental point of view. In addition, the LB method generally needs nearest neighbor information at most so that it is ideally suited for parallel computers. While LB methods are developing rapidly in response to recent theoretical advances and the availability of resources for large scale computation, there is still a lack of critical comparisons between experimental results and simulation. Such comparisons are crucial, not only to validate LB methods, but to further their development. In this paper we examine the utility of the Shan and Chen[1] model of multicomponent fluids for describing large scale flow in complex geometries. This model has been adapted to three dimensions and extended to include fluid-solid interactions and applied forces.[2] After a brief review of the theory of the LB method, results are presented to validate predictions of fluid flow through a few simple pore geometries. Large scale simulations of fluid flow through a Fontainebleau sandstone microstructure, which was generated by X-ray microtomography, will then be presented. Single phase flow calculations were carried out on 510^3 systems. We also calculate relative permeability curves as a function of fluid saturation and driving force. The Onsager relation, which equates off-diagonal components of the permeability tensor for two phase flow, is found not to hold

Further author information:
N.S.M.: E-mail: nicos.martys@nist.gov
J.G.H.: E-mail: john.hagedorn@nist.gov
J.E.D.: E-mail: judith.devaney@nist.gov

for intermediate to low nonwetting saturation as the flow response to an applied body force was nonlinear. Values of relative permeability from three phase flows were compared to corresponding two phase values. Finally, a comparison of the performance of such codes on different computing platforms is given.

2. LATTICE BOLTZMANN MODEL WITH FLUID PHASE SEPARATION

The LB method of modeling fluid dynamics is actually a family[3] of models with varying degrees of faithfulness to the properties of real liquids. These methods are currently in a state of evolution as the models become better understood and corrected for various deficiencies. In this paper we utilize a version of LB proposed by Shan and Chen[1,2] that is particularly simple in form and adaptable to complex flow conditions like the presence of solid-fluid and fluid-fluid boundaries.

The approach of LB is to consider a typical volume element of fluid to be composed of a collection of particles that are represented in terms of a particle velocity distribution function at each point in space. The particle velocity distribution, $n_a^i(\mathbf{x}, t)$, is the number density of particles at node $\mathbf{x}$, time t, and velocity, $\mathbf{e}_a$, where $(a = 1, ..., b)$ indicates the velocity direction and superscript i labels the fluid component. The time is counted in discrete time steps, and the fluid particles can collide with each other as they move under applied forces.

For this study we use the D3Q19 (3 Dimensional lattice with $b = 19$)[4] lattice.[2] The microscopic velocity, $\mathbf{e}_a$, equals all permutations of $(\pm1, \pm1, 0)$ for $1 \leq a \leq 12$, $(\pm1, 0, 0)$ for $13 \leq a \leq 18$, and $(0, 0, 0)$ for $a = 19$. The units of $\mathbf{e}_a$ are the lattice constant divided by the time step. Macroscopic quantities such as the density, $n^i(\mathbf{x}, t)$, and the fluid velocity, $\mathbf{u^i}$, of each fluid component, i, are obtained by taking suitable moment sums of $n_a^i(\mathbf{x}, t)$. Note that while the velocity distribution function is defined only over a discrete set of velocities, the actual macroscopic velocity field of the fluid is continuous.

The time evolution of the particle velocity distribution function satisfies the following LB equation:

$$n_a^i(\mathbf{x} + \mathbf{e}_a, t + 1) - n_a^i(\mathbf{x}, t) = \Omega_a^i(\mathbf{x}, t), \tag{1}$$

where Ω_a^i is the collision operator representing the rate of change of the particle distribution due to collisions. The collision operator is greatly simplified by use of the single time relaxation approximation[5,6]

$$\Omega_a^i(\mathbf{x}, t) = -\frac{1}{\tau_i}\left[n_a^i(\mathbf{x}, t) - n_a^{i(eq)}(\mathbf{x}, t)\right], \tag{2}$$

where $n_a^{i(eq)}(\mathbf{x}, t)$ is the equilibrium distribution at $(\mathbf{x}, t)$ and τ_i is the relaxation time that controls the rate of approach to equilibrium. The equilibrium distribution can be represented in the following form for particles of each type[2,6]:

$$n_a^{i(eq)}(\mathbf{x}) = t_a n^i(\mathbf{x})\left[\frac{3}{2}(1 - d_o) + 3\mathbf{e}_a \cdot \mathbf{v} + \frac{3}{2}(3\mathbf{e}_a\mathbf{e}_a : \mathbf{vv} - \mathbf{v}^2)\right] \tag{3}$$

$$n_{19}^{i(eq)}(\mathbf{x}) = t_{19} n^i(\mathbf{x})\left[3d_o - \frac{3}{2}\mathbf{v}^2\right], \tag{4}$$

where

$$\mathbf{v} = \frac{\sum_i^S m^i \sum_a n_a^i \mathbf{e}_a/\tau_i}{\sum_i^S m^i n^i(\mathbf{x})/\tau_i}, \tag{5}$$

and where m^i is the molecular mass of the ith component, and $t_a = 1/36$ for $1 \leq a \leq 12$, $t_a = 1/18$ for $13 \leq a \leq 18$ and $t_{19} = 1/3$. The free parameter d_o can be related to an effective temperature, T, for the system by the following moment of the equilibrium distribution:

$$T(\mathbf{x}, t) = \frac{\sum_a n_a^{i(eq)}(\mathbf{x}, t)(\mathbf{e}_a - \mathbf{v})^2}{3n^i(\mathbf{x}, t)}, \tag{6}$$

which results in $T = (1 - d_o)/2$ (we take units such that the Boltzmann constant $k_b = 1$).

It has been shown that the above formalism leads to a velocity field that is a solution of the Navier-Stokes[5] equation with the kinematic viscosity, $\nu = \frac{c^2}{6}(\sum_i^S c_i\tau_i - \frac{1}{2})$ where c_i is the concentration of each component.[6]

2.1. Interaction Potential

In order to model the phase separation of fluids, an interaction between the fluids is needed to drive them apart. Here a force, $\frac{d\mathbf{p}^i}{dt}(\mathbf{x})$, between the two fluids is introduced that effectively perturbs the equilibrium velocity[1,2] for each fluid so that they have a tendency to phase separate:

$$n^i(\mathbf{x})\mathbf{v}^{'}(\mathbf{x}) = n^i\mathbf{v}(\mathbf{x}) + \tau_i \frac{d\mathbf{p}^i}{dt}(\mathbf{x}) \tag{7}$$

where $\mathbf{v}'$ is the new velocity used in Eqs. [3] and [4]. We use a simple interaction that depends on the density of each fluid, as follows[1,2]:

$$\frac{d\mathbf{p}^i}{dt}(\mathbf{x}) = -n^i(\mathbf{x}) \sum_{i'}^{S} \sum_{a} G^a_{ii'} n^{i'}(\mathbf{x} + \mathbf{e}_a)\mathbf{e}_a \tag{8}$$

with $G^a_{ii'} = 2G$ for $|\mathbf{e}^a| = 1$;$G^a_{ii'} = G$ for $|\mathbf{e}^a| = \sqrt{2}$; and $G^a_{ii'} = 0$ for $i = i'$. G is a constant that controls the strength of the interaction. Clearly, the forcing term is related to the density gradient of the fluid. It has been shown that the above forcing term can drive the phase separation process and naturally produce an interfacial surface tension effect consistent with the Laplace law boundary condition [3].

In this model, phase separation takes place when the mutual diffusivity of the binary mixture becomes negative. An analytical expression for the mutual diffusivity has been determined in a previous work.[6] For the case of a critical composition the condition for the system studied to undergo phase separation is $G \geq \frac{T}{12(n^1+n^2)}$.

3. IMPLEMENTATION

The approach to implementation of the algorithm is relatively straightforward. At each active site we hold the necessary velocity and mass data for each fluid component. Over the course of an iteration we visit each cell in the data volume and calculate the distribution of each fluid component to be streamed to neighboring cells. New mass and velocity values are accumulated at each cell as its neighbors make their contributions. The most notable aspects of the implementation were our tactics for managing the large amounts of memory required by the algorithm, and the adaptation of the code for use in parallel computing environments.

3.1. Memory Optimizations

Experience with the implementation of related algorithms indicated that the memory required for modeling large systems would be prohibitive. We therefore looked for ways to conserve and reduce memory usage. There are several tactics that we used in this implementation:

- Store data only at the active sites.
  This is accomplished in the C implementation by representing the medium as a three dimensional array of pointers. At each active site the pointer references a data structure with the necessary velocity and mass data. At the inactive sites the pointer is NULL; no additional storage is required at the inactive sites. For a low porosity medium the memory savings are very large.
- Assume that $\tau = 1$.
  This assumption simplifies evaluation of equations 1-5 such that at each active site we need only store the density of each fluid component, and a single velocity vector. Without this assumption, we must store all 19 values associated with the velocity distribution, n_i, at each site.
- Only one copy of the data volume is stored.
  Rather than keeping an entire second data volume in which to accumulate the newly calculated data, we exploit the fact that the algorithm only uses nearest neighbors at each site. Thus we only need an additional buffer of three planes of data at any one time.

Assuming that floating point numbers and C pointers each take four bytes, these memory optimizations yield savings of over 94 % of memory usage in the one component case for systems of useful sizes. The memory savings are even greater when more fluid components are used or when larger floating point representations are used.

3.2. Parallelization

The amount of computation and memory required for a large system suggested that it would be advantageous to adapt the implementation so that a single problem could be run in parallel across a collection of processors. The nearest-neighbor dependence of the algorithm also suggested that parallelization would be straightforward and would yield substantial benefits. Parallelization enables us to run larger systems by distributing the memory requirements across many machines, and gives us faster performance by distributing the computation.

We implemented the parallel version of the algorithm using the Message Passing Interface[7] (MPI). This is an industry-standard library of routines for coordinating execution and communicating between processes in a parallel computing environment. The parallelization was accomplished within a simple Single Program Multiple Data (SPMD) model. The data volume is divided into spatially contiguous blocks along the Z axis; multiple copies of the same program run simultaneously, each operating on its block of data. Each copy of the program runs as an independent process and typically each process runs on its own processor. At the end of each iteration, data for the planes that lie on the boundaries between blocks are passed between the appropriate processes and the iteration is completed. The periodic boundary condition is handled transparently; the process handling the "top" plane of data volume simply exchanges data with the process handling the "bottom" plane of the data volume.

4. NUMERICAL TESTS

Several numerical tests were carried out to verify our algorithm. Results from two cases, fluid flow between parallel plates and through an overlapping sphere model, are given below. For both cases we determined the fluid permeability, k, as defined by Darcy's law, $\langle \vec{v} \rangle = -\frac{k}{\mu}\nabla P$, where $\langle \vec{v} \rangle$ is the average flow rate, ∇P is the pressure gradient and μ is the fluid viscosity. Figure 1 shows the permeability, in units of the lattice spacing squared, as a function of the distance between parallel plates. Clearly, there is excellent agreement between the simulation and theoretical prediction. Surprisingly, very accurate results were obtained even for the case of a one node wide channel. Since permeability depends on the average flow or net flux rate of fluid, we conclude that the LB method accurately determines the net flux across a voxel surface, not the velocity at a point. Hence, resolving the actual local flow field at a point would require more nodes. We next consider the permeability of a simple cubic array of spheres that are allowed to overlap for large enough radius (i.e. when the solid fraction, c, exceeds $c \approx .5236$). In Fig. 2 we compare our simulation data with that of Chapman and Higdon,[8] which is based on the numerical solution of coefficients of a harmonic expansion that satisfies the Stokes equations. Note that our calculations were performed on a relatively small 64^3 system. Again, agreement is very good, especially given that the solid inclusion is a digitized sphere.

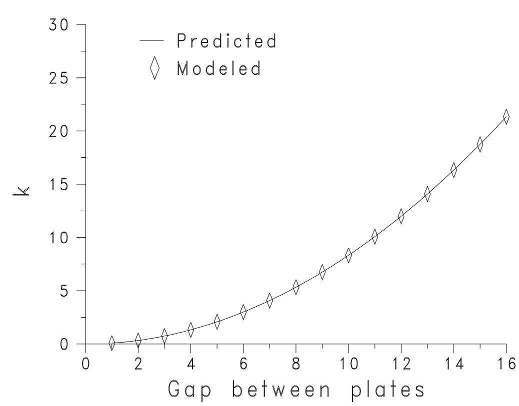

Figure 1. Flow through parallel plates.

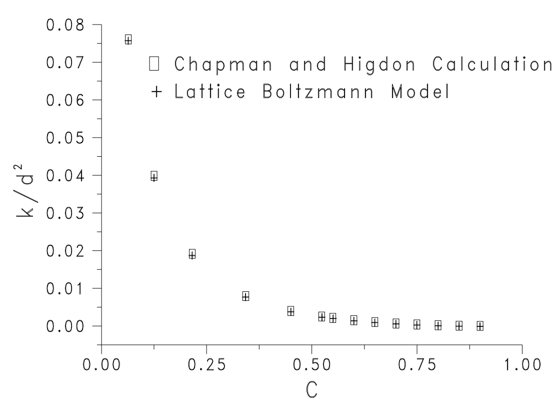

Figure 2. Flow through spheres centered on a simple cubic lattice. The permeability is normalized by the square of the distance, d, between sphere centers.

5. COMPARISON WITH EXPERIMENTAL DATA

We next determined the permeability of several microtomography-based images of Fontainebleau sandstone. Figure 3 depicts portions of two of these sandstone images. The resolution was 5.72μm per lattice spacing and data sets were 510^3 voxels. A mirror image boundary condition was applied along directions perpendicular to the applied forcing. The porous medium was made periodic in the flow direction by creating its mirror image at the inlet. The numerical calculations were carried out on a $1020 \times 510 \times 510$ system for all but the lowest porosity system. We found that at the lowest porosity (7.5 %) there were not enough nodes across the pores to produce a reliable flow field. So for this case the permeability was determined from a 256^3 piece of the sandstone image that was mapped to a 512^3 image, and calculations were performed on a $1024 \times 512 \times 512$ system. In addition to requiring sufficient resolution, another potential source of error is not having precise knowledge of the location of the pore/solid interface. For example, an error of half a lattice spacing could be significant when modeling flow in narrow channels like that in the low porosity system. Figure 4 shows the computed permeability compared to experimental data.[9] Clearly there is good agreement, especially at the higher porosities.

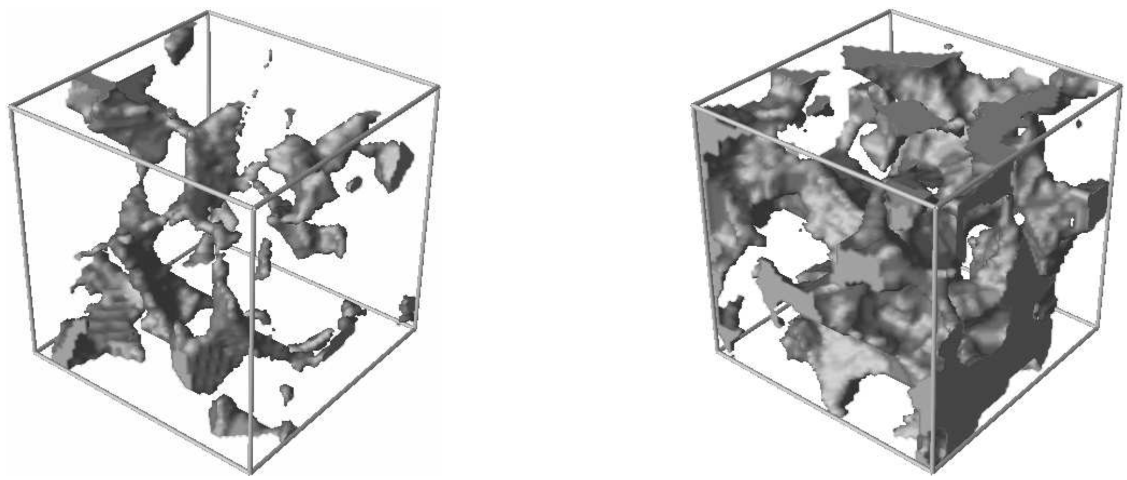

Figure 3. 64×64 portions of the Fontainebleau sandstone media. On the left is the 7.5 % porosity medium, on the right is the 22 % porosity medium. The solid matrix is made transparent to reveal the pore space (grey shaded region).

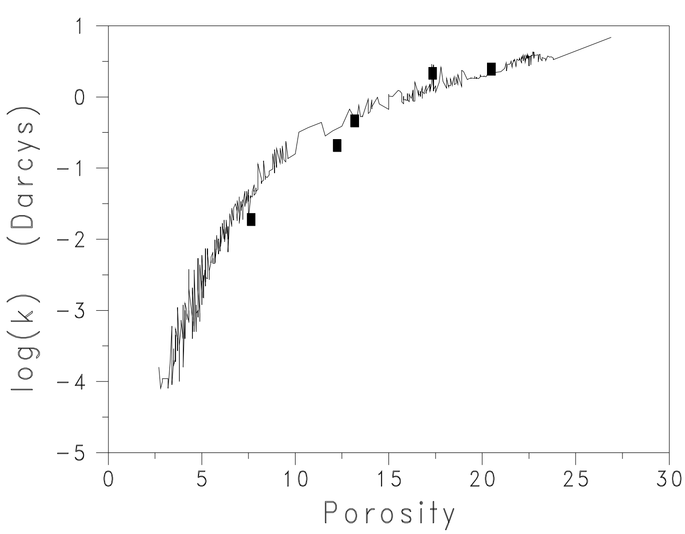

Figure 4. Measured and modeled permeabilities of Fontainebleau sandstone medium.

6. RELATIVE PERMEABILITY

We next present a sample calculation of the relative permeability for the 22 % porosity Fontainebleau sandstone. Although there is debate as to the correct formulation of the macroscopic two phase flow equations,[10] we use the following empirical relation to describe the response of a multiphase fluid system to an external driving force:

$$\vec{v}_1 = -\frac{K_{12}}{\mu_2}\nabla P_2 - \frac{K_{11}}{\mu_1}\nabla P_1 \tag{9}$$

$$\vec{v}_2 = -\frac{K_{21}}{\mu_1}\nabla P_1 - \frac{K_{22}}{\mu_2}\nabla P_2 \tag{10}$$

Here the K_{ij} are the components of a permeability tensor and the applied pressure gradient on each fluid component ∇P_i is from a simple body force, $\nabla P = \rho g$, where g is an acceleration constant. The forcing can be applied to each phase separately allowing determination of the off-diagonal terms in the permeability tensor. The viscosity μ_i is the same for both fluids. Relative permeability data is usually presented in terms of constant capillary number, $C_a = \frac{\mu v}{\gamma}$, where γ is the interfacial surface tension. For our body force driven fluids, we can define an effective capillary number, C_a^*, by replacing v with the Darcy velocity so that $C_a^* = \frac{\mu < v >}{\gamma} = \frac{k \rho g}{\gamma}$. Below is a plot of the relative permeability of the $\phi = 22$ % rock for the cases of $C_a^* = 7.5 \times 10^{-4}$ and 7.5×10^{-5}.

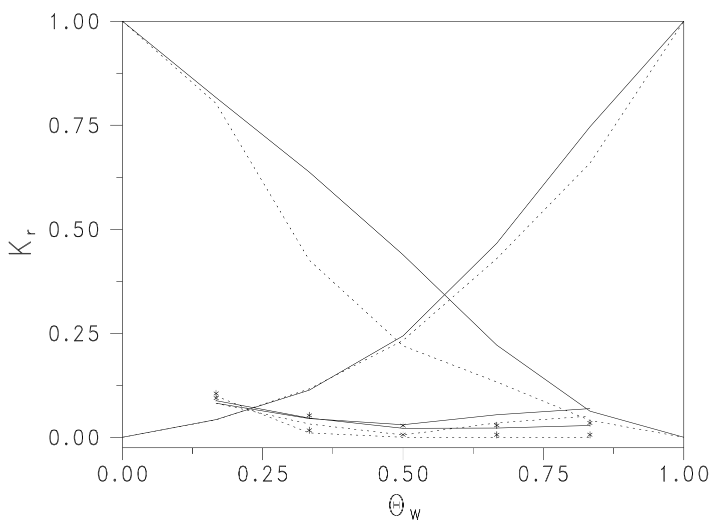

Figure 5. Relative permeabilities of 22 % porosity Fontainebleau sandstone versus wetting fluid saturation, Θ_W. The solid and dashed lines correspond to $C_a^* = 7.5 \times 10^{-4}$ and $C_a^* = 7.5 \times 10^{-5}$ respectively. The lower curves correspond to the off-diagonal elements of the permeability tensor with the $*$ denoting the case where the nonwetting fluid is driven.

Clearly, as the forcing decreases the relative permeability decreases. Also note that at lower wetting saturation the relation $K_{12} = K_{21}$ holds fairly well. This is the well known Onsager relation. However, as the wetting phase increases, this relation appears to break down. In this regime, the nonwetting phase is beginning to form disconnected blobs that do not respond in a linear fashion to the applied force due to pinning effects as the nonwetting blobs are pushed through the smaller pores.

The LB code can be easily extended to model three or more fluid components. As a simple test case we considered a three component system with each component having a pore volume fraction of 1/3. In addition, two phases were made non-wetting while the third was wetting. A forcing was applied to one of the nonwetting phases. We found, over the range $7.5 \times 10^{-4} < C_a^* < 3.75 \times 10^{-3}$, there was an approximately 30 % to 35 % decrease in flow of the driven nonwetting fluid as compared to the case where 1/3 fluid was non wetting and 2/3 was wetting. Presumably, this decrease can be understood as the non-driven nonwetting phase interfering with the flow of the driven nonwetting phase as the fluids move through narrow channels.

7. PERFORMANCE RESULTS

We ran a series of timing tests in an effort to understand how performance of our implementation scales on different computer architectures. We have tested on an SGI Onyx with 12 R10000 processors running at 196MHz, an IBM SP2 with 37 RS/6000 processors, most running at 66MHz. The same code and the same cases were run on the two systems. The results are presented in Tables 1 and 2. The performance reported was somewhat affected by other jobs that were running at the same time that the tests were being run, although efforts were made to minimize the effect.

# Processors	# Fluid Components 1	2	3
1	14.70	24.70	33.27
2	7.39	12.22	16.69
4	3.80	6.23	8.57
8	2.14	3.48	4.68

Table 1. Execution times in seconds for one iteration on the SGI Onyx.

# Processors	# Fluid Components 1	2	3
1	38.48	62.36	99.93
2	19.30	31.32	51.16
4	10.44	16.83	26.97
8	6.86	10.01	15.54
16	4.37	6.00	8.30

Table 2. Execution time in seconds for one iteration on the IBM SP2.

These data closely agree with a very simple model describing performance: $T = P/N + S$, where T is the total time for a single iteration, P is the time for the parallelizable computation, S is the time for the non-parallelizable computation, and N is the number of processors. The parallelizable computation is that portion of the processing that can be effectively distributed across the processors. The non-parallelizable computation includes processing that cannot be distributed; this includes time for inter-process communication as well as computation that must be performed either on a single processor, or must be done identically on all processors.

For example, the two-component fluid performance data for the SGI Onyx, closely match this formula: $T = 4.78 + 487.26/N$ seconds, where N is the number of processors. Similarly, the timings for the two component runs on the IBM SP2 closely match: $T = 41.67 + 1198.45/N$ seconds. Formulae for the other cases are easily derived. Figures 6 and 7 present these results graphically.

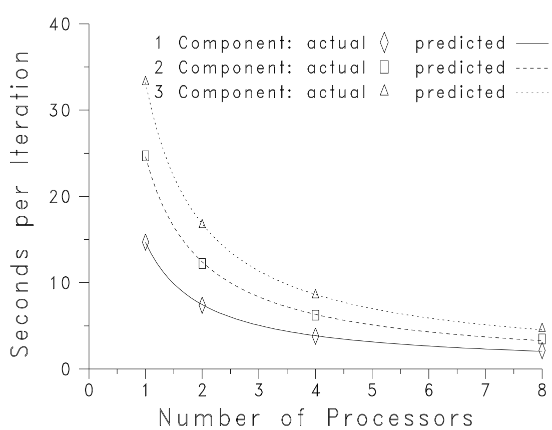

Figure 6. Time in seconds for one iteration on the SGI Onyx.

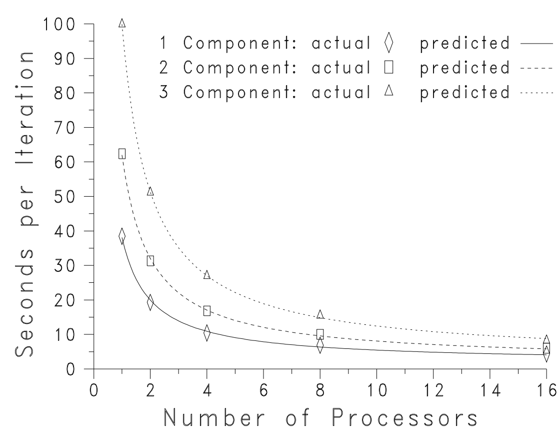

Figure 7. Time in seconds for one iteration on the IBM SP2.

Much of the difference between the performance of these two systems is likely due simply to the relative computational speeds of each processor. But the difference in the serial overhead (4.78 seconds on the SGI versus 41.67

seconds on the IBM), is most likely due to the different memory architectures of the two systems. The SGI Onyx uses a Non-Uniform Memory Access (NUMA) architecture that enables processes to pass data to one another through shared memory, However, on the IBM SP2 no memory is shared and data must be transferred over an external high-speed network. Thus the overhead for message passing on the SGI Onyx is considerably lower than that on the IBM SP2. We intend to run timing tests to measure the difference in message passing overhead.

The time for the parallelizable portion of the code is expected to be in proportion to the number of active sites, which depends on the porosity and the size of the volume. But the time for the non-parallelizable portion of the code is likely to be dominated by the inter-process communication. Assuming that communication time is roughly proportional to the amount of data transferred, the communication time should be proportional to the number of active sites on an XY plane.

So as we process larger systems, the time for the parallelizable portion of the code should increase proportionally with the cube of the linear size of the system, while the non-parallelizable portion should increase with the square of the linear size of the system. This means that for larger systems, a larger proportion of the time is in the parallelizable computation, and greater benefits can be derived from running on multiple processors. We are still in the process of investigating the scaling of the software's performance with system size.

These performance data give us a general idea of how long it takes to get practical results for real-world problems on the computing platforms tested. For example, a typical case requires about 10000 iterations to converge. So from the performance described above, a one-component run of the sample size and porosity (22 %) described above will take about 41 hours on one processor on an SGI Onyx. On four processors, the same run will take approximately 10.6 hours. Approximate times for other sizes and porosities are easily calculated from the data above.

8. CONCLUSION

Lattice Boltzmann methods for simulating fluid flow in complex geometries have developed rapidly in recent years. The LB method produces accurate flows and can accommodate a variety of boundary conditions associated with fluid-fluid and fluid-solid interactions. With the advent of large memory/parallel workstations (or PC clusters), computations on fairly large systems that were considered beyond the reach of even some "super" computers from a decade ago can now be considered routine. Further work is needed to model fluids with large viscosity and density mismatches, which is the subject of current research.

ACKNOWLEDGMENTS

Fontainebleau sandstone images were prepared by John Dunsmuir of Exxon Research & Engineering Co. in collaboration with Brent Lindquist and Teng-Fong Wong (SUNYSB) at the National Synchrotron Light Source, Brookhaven National Laboratory, which is supported by the U.S. Department of Energy, Division of Materials Sciences and Division of Chemical Sciences under contract number DE-AC02-98CH10886.

DISCLAIMER

Certain commercial equipment and software may be identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment or software is necessarily the best available for the purpose.

REFERENCES

1. X. Shan and H. Chen, "A lattice Boltzmann model for simulating flows with multiple phases and components," *Phys. Rev. E* **47**, pp. 1815–1819, 1993.
2. N. S. Martys and H. Chen, "Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice Boltzmann method," *Phys. Rev. E* **53**, pp. 743–750, 1996.
3. D. H. Rothman and S. Zaleski, "Lattice-gas model of phase separation: interfaces, phase transitions, and multiphase flow," *Rev. Mod. Phys.* **66**(4), pp. 1417–1479 and references, 1998.
4. Y. H. Qian, D. d'Humières, and P. Lallemand, "Lattice bgk models for Navier-Stokes equation," *Europhys. Lett.* **17**, pp. 479–484, 1992.

5. H. Chen, S. Y. Chen, and W. H. Matthaeus, "Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method," *Phys. Rev. A* **45**, pp. R5339–R5342, 1992.
6. X. Shan and G. Doolen, "Diffusion in a multicomponent lattice Boltzmann equation model," *Phys. Rev. E* **54**, pp. 3616–3620, 1996.
7. M. P. I. Forum, "MPI: A message-passing interface standard," *International Journal of Supercomputing Applications* **8**(3/4), 1994.
8. A. M. Chapman and J. J. L. Higdon, "Oscillatory Stokes flow in periodic porous media," *Phys. Fluids A* **4**(10), pp. 2099–2116, 1992.
9. T. Bourbie and B. Zinszner, "Hydraulic and acoustic properties as a function of porosity in Fontainebleau sandstone," *J. Geophys. Res.* **90**(B13), pp. 11,524–11,532, 1985.
10. P. A. Goode and T. Ramakrishnan, "Momentum transfer across fluid-fluid interfaces in porous media: a network model," *AIChE Journal* **39**(7), pp. 1124–1134, 1993.