International Journal of Modeling, Simulation,
and Scientific Computing
Vol. 7, No. 3 (2016) 1650009 (15 pages)

DOI: 10.1142/S1793962316500094

DSMC: Fast direct simulation Monte Carlo solver for the Boltzmann equation by Multi-Chain Markov Chain and multicore programming

Di Zhao* and Haiwu He

Computer Network Information Center
Chinese Academy of Sciences
Beijing 100190, P. R. China
**zhaodi@sccas.cn*

Received 19 June 2015
Accepted 29 December 2015
Published 17 February 2016

Direct Simulation Monte Carlo (DSMC) solves the Boltzmann equation with large Knudsen number. The Boltzmann equation generally consists of three terms: the force term, the diffusion term and the collision term. While the first two terms of the Boltzmann equation can be discretized by numerical methods such as the finite volume method, the third term can be approximated by DSMC, and DSMC simulates the physical behaviors of gas molecules. However, because of the low sampling efficiency of Monte Carlo Simulation in DSMC, this part usually occupies large portion of computational costs to solve the Boltzmann equation. In this paper, by Markov Chain Monte Carlo (MCMC) and multicore programming, we develop Direct Simulation Multi-Chain Markov Chain Monte Carlo (DSMC3): a fast solver to calculate the numerical solution for the Boltzmann equation. Computational results show that DSMC3 is significantly faster than the conventional method DSMC.

Keywords: Fast solver; direct simulation Multi-Chain Markov Chain Monte Carlo; DSMC; the Boltzmann equation; multicore programming.

1. Introduction

Mathematical description of gas dynamics can be built by continuous models and discrete models. While continuous models mainly include Euler equation and Navier–Stokes equation, discrete models include Boltzmann equation and collisionless Boltzmann equation,[1] and the solving methods for these equations include analytical methods and numerical methods.

Numerical methods for gas dynamics models are usually expensive. Gas dynamics models such as Navier–Stokes equation possess the variable of time, and large

*Corresponding author.

amount of computational costs are spent on the iterations of time progress, and acceleration techniques such as mathematical methods and computational methods are developed.[2–4]

Direct Simulation Monte Carlo (DSMC) applies Monte Carlo Simulation to solve the Boltzmann equation for large Knudsen number fluid.[5] There are four main parts in the algorithm of DSMC: initialization, sorting, collision and post-processing. Comprehensive research of DSMC is conducted for decades: collision model,[6] particle resolution,[7] the number of collision,[8] collision sampling algorithm,[9,10] convergence[11] and error analysis.[12] Since DSMC is a numerical solution method for the Boltzmann equation, the main application areas of DSMC comprise flow simulation,[13–30] heat transfer,[31] etc.

Monte Carlo Simulation based algorithms usually need intensive computation, and DSMC is a typical example to illustrate the demands. Therefore, existing research for accelerating DSMC such as CPU implementation[32–36] and GPU implementation[37] is conducted. However, solving large-scale problems such as fluid simulation in complicated geometry still leads to large amount of collision sampling, which makes the computational cost of the conventional DSMC high.

While Monte Carlo Simulation uses uniform distribution for approximating the target distribution of the collisions among the gas particles, samplers such as Markov Chain Monte Carlo (MCMC) can be employed for higher efficiency, where MCMC is a sampling algorithm to construct the approximation of the target distribution by Markov Chain, and MCMC is usually more efficient than Monte Carlo Simulation under multiple circumstances such as genomical computation[38] Li developed fast algorithms to solve the Boltzmann equation.[39–41]

In this paper, by MCMC and multicore vectorization, we develop Direct Simulation Multi-Chain Markov Chain Monte Carlo (DSMC3): a fast solver to calculate the numerical solution for the Boltzmann equation. The numerical and computational details of DSMC3 are discussed, and the computational results are presented to illustrate the performance of this solver.

The contribution of this paper is that we develop DSMC3, a multicore vectorization-based Direct Simulation Multi-Chain Markov Chain Monte Carlo solver, which is much faster than the conventional method DSMC.

2. Methods

In this section, we firstly introduce the Boltzmann equation for the dilute gas. Then we introduce the conventional method DSMC, the existing numerical solvers for the Boltzmann equation. To increase the efficiency of DSMC, we firstly develop Direct Simulation Markov Chain Monte Carlo (DSMC2), and we then develop DSMC3.

2.1. *The Boltzmann equation*

The Boltzmann equation is a discrete model to describe the statistical behavior of gas molecular with large Knudsen number. The Boltzmann equation generally

holds three terms: the force term, the diffusion term and the collision term. The Boltzmann equation can be described by the general format of Eq. (1a):

$$\frac{\partial F}{\partial t} + v \cdot \nabla_x F = Q(F, F), \tag{1a}$$

where the variable $F(t, x, v)$ is the probability density function for the statistical behaviors of gas particles, and the collision term is a complicated integration:

$$Q[F(v), F(v)] = \int_{R^3} dv_* \int_{S^2} d\sigma(|v - v_*|, \sigma)[F(v'_*)F(v') - F(v_*)F(v)]. \tag{1b}$$

Since directly calculating the integration of Eq. (1b) is difficult, approximation methods such as Monte Carlo Simulation can be applied to calculate the integration.

2.2. *DSMC*: *Direct Simulation Monte Carlo*

DSMC is an algorithm of applying Monte Carlo Simulation to solve the Boltzmann equation. The main idea of DSMC is representing the dilute gas by particles, dividing the particles of the dilute gas into manually defined cells, and conducting the physical activities such as collision within these cells. DSMC describes molecular behaviors such as collision of the dilute gas in the unit of particle, which is the collection of large number of gas atoms.

To solve the collision term in the Boltzmann equation of Eq. (1b), DSMC randomly selects two particles from the cell by the uniform distribution, and a logical comparison is calculated:

$$U(0, v_{\max}) \leq v(p_1, p_2). \tag{2}$$

If the logic value of Eq. (2) is true, the collision function

$$\beta(p_1, p_2) \tag{3}$$

is calculated, where the collision function of Eq. (3) generally possesses the Brown motion, the gravity and the turbulence. The details of DSMC are described in Algorithm 1:

Algorithm 1. Direct Simulation Monte Carlo (DSMC)

Initialization: create the particles by the boundary conditions;

WHILE $t < T$
    FOR all cells
        Sorting: sort particles into selected number of cells;
        Collision: calculate inner-cell collisions by Eqs. (2) and (3);
    END FOR
END WHILE
Post-processing.

From Algorithm 1, we can see that the DSMC (Algorithm 1) consists of two loops: the WHILE loop for time progress and the FOR loop for all cells. In every

cell at every iteration for time progress, the steps of sorting and collision are calculated. Since the sampler of Eq. (2) obeys the uniform distribution, the efficiency of this sampler is usually low, which leads to large number of calculating the collision function of Eq. (3). To increase the performance of the sampler, DSMC2 is developed.

2.3. *DSMC2: Direct Simulation Markov Chain Monte Carlo*

As described in DSMC (Algorithm 1), the sampler obeys the uniform distribution, and the algorithm does not take advantage of the information from the previous samples. To increase the efficiency of the sampling, the sampler can be built by Markov Chain as:

$$u(0,1) \leq \frac{v(p_1^n, p_2^n)}{v(p_1^{n-1}, p_2^{n-1})}, \tag{4}$$

where $p^{n-1}$is the sample of the previous step.[9] The set of the sampled collision pairs by equation (4) constructs Markov Chain. The details of DSMC2 are described in Algorithm 2:

Algorithm 2. Direct Simulation Markov Chain Monte Carlo (DSMC2)

Initialization: create the particles by the boundary conditions;

WHILE $t < T$
    FOR all cells
        Update the positions of all particles;
        Sorting: sort particles into selected number of cells;
        Collision: calculate inner-cell collisions by Eqs. (4) and (3);
            Propose a collision pair (p_1^n, p_2^n);
            Calculate Eq. (4): if the logical value is true, calculate the collision function of Eq. (3); if the logical value if false, re-propose the collision pair (p_n^1, p_n^2);
    END FOR
END WHILE
Post-processing.

From Algorithm 2 we can see, DSMC2 (Algorithm 2) also consists of two loops: the WHILE loop for time progress and the FOR loop for all cells. In Algorithm 2, the collision pairs are sampled by Eq. (4), and the collision of the pair is calculated by Eq. (3). Since the sampling efficiency from Markov Chain by Eq. (4) is higher than the uniform distribution by Eq. (2), the number of calculating the collision term of Eq. (3) in DSMC2 (Algorithm 2) is lower than DSMC (Algorithm 1), which is also shown in the next section of Computational Results.

In DSMC2 (Algorithm 2), every cell contains an independent Markov Chain, and there is no cross-cell information exchange. Since the behaviors of every Markov Chain are different, DSMC2 (Algorithm 2) runs sequentially in every cell. To further

increase the efficiency by redesigning DSMC (Algorithm 2) and parallelizing Markov Chain by vectorization, we develop Direct Simulation Multi-Chain Markov Chain Monte Carlo (DSMC3).

2.4. *DSMC3: Direct Simulation Multi-Chain Markov Chain Monte Carlo*

Multi-Chain Markov Chain is a variant of Markov Chain that contains multiple chains. To satisfy the Single Instruction Multiple Data (SIMD) requirements of multicore vectorization, by the new design from DSMC2 (Algorithm 2) and multicore vectorization, we develop DSMC3 with the sampler:

$$U(0,1) \leq \frac{v(P_1^n, P_2^n)}{v(P_1^{n-1}, P_2^{n-1})}, \tag{5}$$

where the vector P^{n-1} is the samples of the previous step. The details of DSMC3 are shown in Algorithm 3:

Algorithm 3. Direct Simulation Multi-Chain Markov Chain Monte Carlo (DSMC3)

Initialization: create the particles by the boundary conditions;

WHILE $t < T$

Update the positions of all particles;

Sorting: sort particles into the selected number of cells;

Collision: calculate inner-cell collisions by Eqs. (4) and (3);

Propose the vector of the collision pairs (P_n^1, P_n^2) for all cells;

Calculate the collision function of Eq. (3) for all cells;

Calculate Eq. (4) for all cells: for every cell, if the logical value is true, update the calculated velocity; if the logical value is false, keep the velocity the same;

END

Post-processing.

Comparing DSMC3 (Algorithm 3) against DSMC2 (Algorithm 2), the design of the collision part is different: while DSMC2 (Algorithm 2) and DSMC (Algorithm 1) consist of two loops, DSMC3 (Algorithm 3) only contains a WHILE loop. In DSMC2 (Algorithm 2), the collision function of Eq. (3) is calculated if Eq. (4) is true; in DSMC3 (Algorithm 3), the collision function of Eq. (3) is calculated under any circumstance, but the calculated velocity is updated only if Eq. (4) is true. The framework of DSMC3 (Algorithm 3) is also shown in Fig. 1.

From Fig. 1 we can see, every cell contains Markov Chain, and these chains share a common structure: the same implementation of Algorithm 3 and the same chain length. In Fig. 1, the dotted rectangle means the vector (P_n^1, P_n^2) in Algorithm 3. These chains are independent in sampling particle pairs within cells, and every chain constructs an independent approximation for the cell. By the framework in Fig. 1,

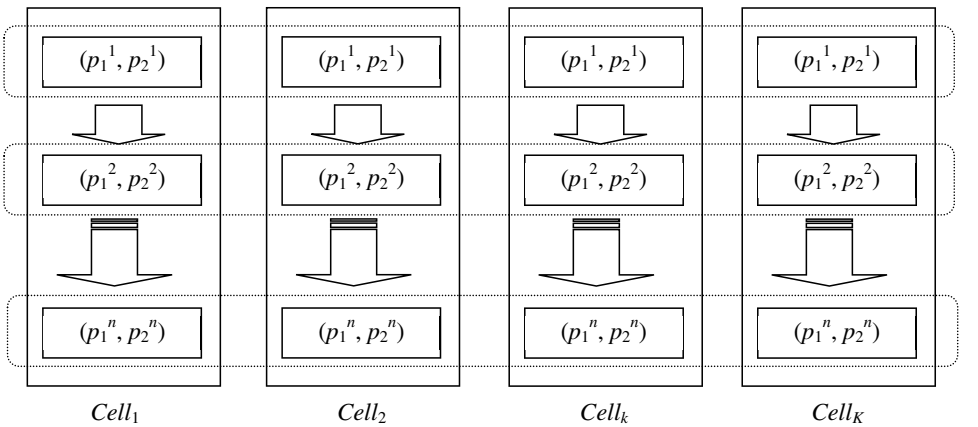

Fig. 1. The framework of Direct Simulation Multi-Chain Markov Chain Monte Carlo (DSMC3).

DSMC3 (Algorithm 3) is with the style of SIMD, which results in high efficiency in multicore vectorization.

3. Computational Results

By referencing the DSMC codes in Ref. 1, we code DSMC3 (Algorithm 3), DSMC2 (Algorithm 2) and DSMC (Algorithm 1) by MATLAB, and we test the three algorithms by the canonical example of gas thermodynamic equilibrium. The computational platform for these experiments is a GPU workstation with Intel i7-2600 K CPU, 32 GB memory, and this CPU contains four cores.

3.1. *The problem*

The numerical example in this section is the gas thermodynamic equilibrium problem,[42] and the dilute gas with argon atom is selected. The gas temperature is set as 300 K.[42] The particles represent a large number of homogeneous gas atoms, which collides each other and collide against the wall.[42] The gas is enclosed in a cuboid with four sides in specular reflection and two sides in periodic boundary condition.[42] That is, by specular reflection, the particles reflect off the wall with the reversed velocity:

$$v_{\text{out}} = -v_{\text{in}}.$$

The wall boundary condition is applied to the numerical example, that is, the normal component and the tangential component of the boundary condition are set to zero:

$$V_{\text{normal}} = 0, \quad V_{\text{tangential}} = V_{\text{wall}}.$$

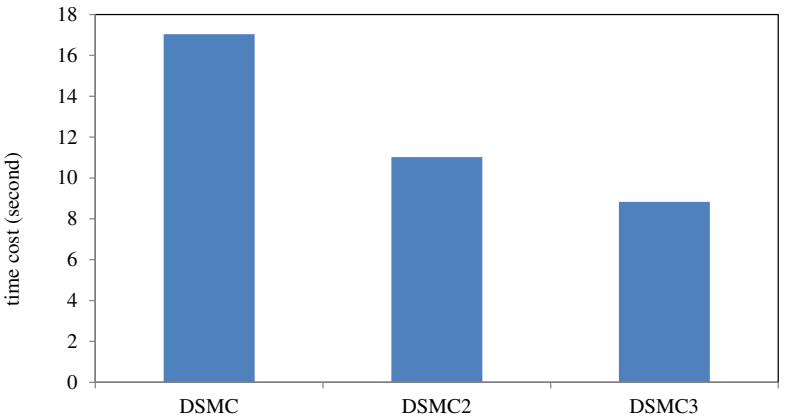

Fig. 2. The comparison among the time cost of DSMC3 (Algorithm 3), DSMC2 (Algorithm 2) and DSMC (Algorithm 1) in seconds.

3.2. *The performance*

The performance of DSMC3 (Algorithm 3) is tested against the conventional method DSMC (Algorithm 1). The number of particles in the simulated gas is set as 500,000. All three algorithms iterate for 20 time steps to reach the equilibrium, and every time step represents 10^{-10} s.

Figure 2 shows the comparison among the time cost of DSMC3 (Algorithm 3), DSMC2 (Algorithm 2) and DSMC (Algorithm 1) in seconds. 10-chain Markov Chain runs for DSMC3 (Algorithm 3) with 8500 states.

From Fig. 2 we can see, the time cost of DSMC2 (Algorithm 2) is significantly lower than DSMC (Algorithm 1). The reason why DSMC2 (Algorithm 2) is faster than DSMC (Algorithm 1) is Markov Chain, the number of calculated collisions by DSMC2 (Algorithm 2) is less than DSMC (Algorithm 1), and the sampling efficiency of Markov Chain of Eq. (4) is higher than the uniform distribution of Eq. (2).

From Fig. 2 we can see, the time cost of DSMC3 (Algorithm 3) is significantly lower than DSMC2 (Algorithm 2) and DSMC (Algorithm 1). The reason why DSMC3 (Algorithm 3) is faster than DSMC2 (Algorithm 2) and DSMC (Algorithm 1) is 10-chain Markov Chain and multicore parallelization, the number of calculated collisions by DSMC3 (Algorithm 3) is much less than DSMC2 (Algorithm 2) and DSMC (Algorithm 1), and multicore parallelization makes multiple chains simultaneously work for highly efficiently sampling.

Figure 3 shows the number of calculated collisions for DSMC3 (Algorithm 3), DSMC2 (Algorithm 2) and DSMC (Algorithm 1) along with time steps, and the total number of time steps is 20. From Fig. 3 we can see, in every time step, the number of calculated collisions of DSMC3 (Algorithm 3) is much smaller than DSMC2 (Algorithm 2) and DSMC (Algorithm 1), which is fixed at 8500.

Does DSMC3 (Algorithm 3) output the same $F(t, x, v)$ to the conventional method DSMC (Algorithm 1)? Figure 4 shows the calculated velocity distribution

of DSMC (Algorithm 1), DSMC2 (Algorithm 2) and DSMC3 (Algorithm 3). From Fig. 4 we can see, the calculated velocity distributions of DSMC (Algorithm 1), DSMC2 (Algorithm 2) and DSMC3 (Algorithm 3) are approximately the same, which coincides with the Maxwell–Boltzmann distribution.

3.3. *The scalability for chain length*

The scalability for the chain length of DSMC3 (Algorithm 3) is also tested. The number of particles in the simulated gas is set as 1,000,000. The two algorithms iterate for 20 time steps to reach the equilibrium, and every time step represents 10^{-10} s. The number of chains is set to 10.

Figure 5 shows the time cost of DSMC3 (Algorithm 3) and DSMC2 (Algorithm 2) along with the chain length. From Fig. 5 we can see, as the chain lengths increase, the time cost of DSMC3 (Algorithm 3) and DSMC2 (Algorithm 2) increases. From Fig. 5 we can also see, the time cost of DSMC3 (Algorithm 3) along with the chain length is much less than that of DSMC2 (Algorithm 2).

How does $F(t, x, v)$ from DSMC3 (Algorithm 3) change along with the chain length? To answer this question, we record the calculated velocity distribution with different chain lengths, and the results are plotted in Fig. 6.

Figure 6(i) shows the situation when the 10-chain DSMC3 with the length of 10,000 states does not converge to the stationary distribution, and the shape of the velocity distribution is far from the Maxwell–Boltzmann distribution. After the chain length of 10,000 states, the 10-chain DSMC3 reach the stationary distribution, and the shapes of the velocity distribution in Figs. 6(ii) to 6(ix) are almost the same.

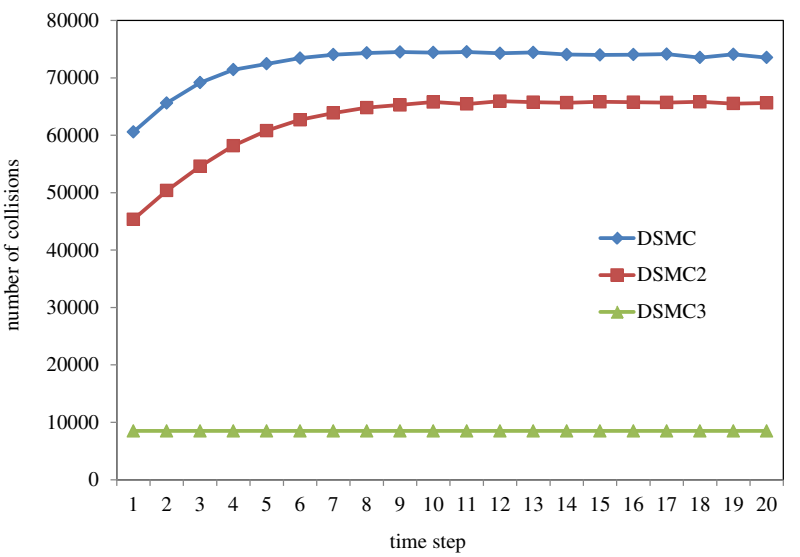

Fig. 3. The number of calculated collisions of DSMC3 (Algorithm 3), DSMC2 (Algorithm 2) and DSMC (Algorithm 1) along with time steps.

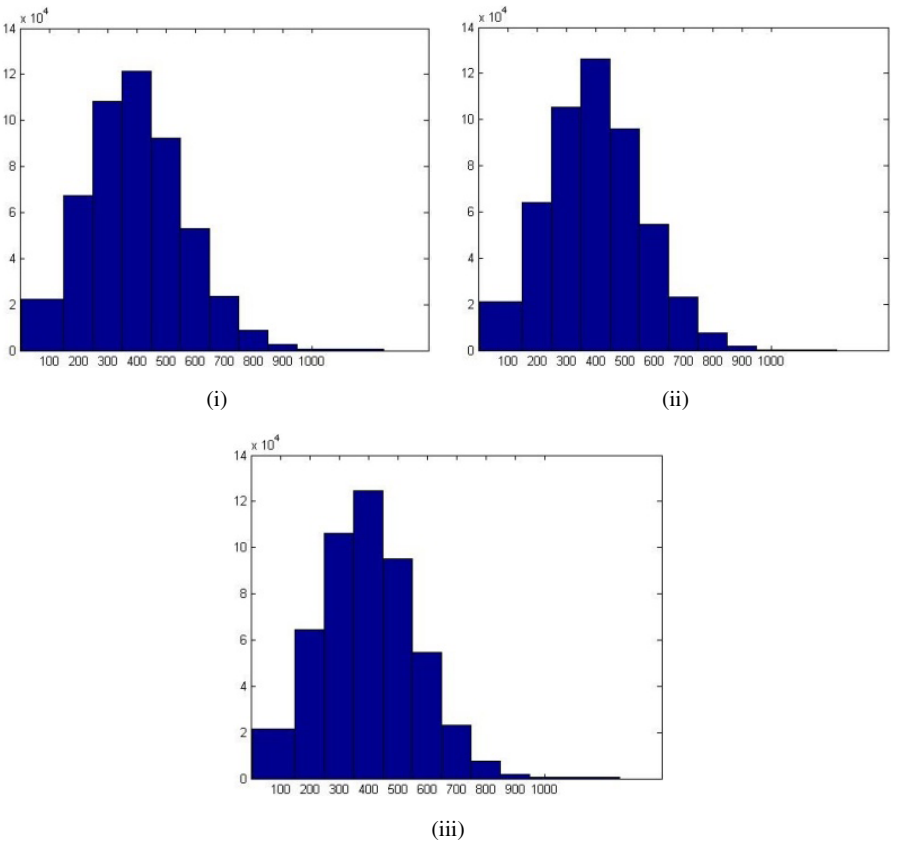

Fig. 4. The velocity distribution of (i) DSMC (Algorithm 1), (ii) DSMC2 (Algorithm 2) and (iii) DSMC3 (Algorithm 3).

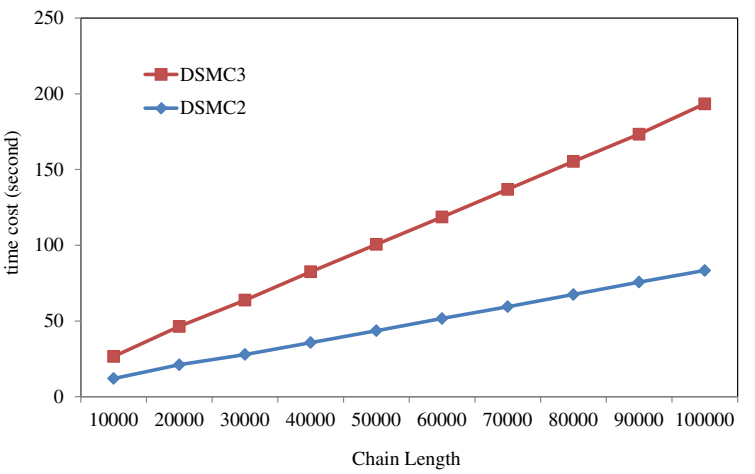

Fig. 5. The time cost of DSMC3 (Algorithm 3) and DSMC2 (Algorithm 2) along with the chain length.

3.4. *The scalability for number of chains*

The scalability for the number of chains is tested for DSMC3 (Algorithm 3). The number of particles in the simulated gas is set as 2,000,000. The chain length of DSMC3 is fixed at 20,000. The two algorithms iterate for 20 time steps to reach the equilibrium, and every time step represents 10^{-10} s.

Figure 7 shows the time cost of DSMC3 (Algorithm 3) and DSMC2 (Algorithm 2) along with the number of chains. From Fig. 7 we can see, as the number of chains increases, the time cost of DSMC3 (Algorithm 3) and DSMC2 (Algorithm 2) increases. From Fig. 7 we can also see, the time cost of DSMC3 (Algorithm 3) along with the number of chains is much less than that of DSMC2 (Algorithm 2).

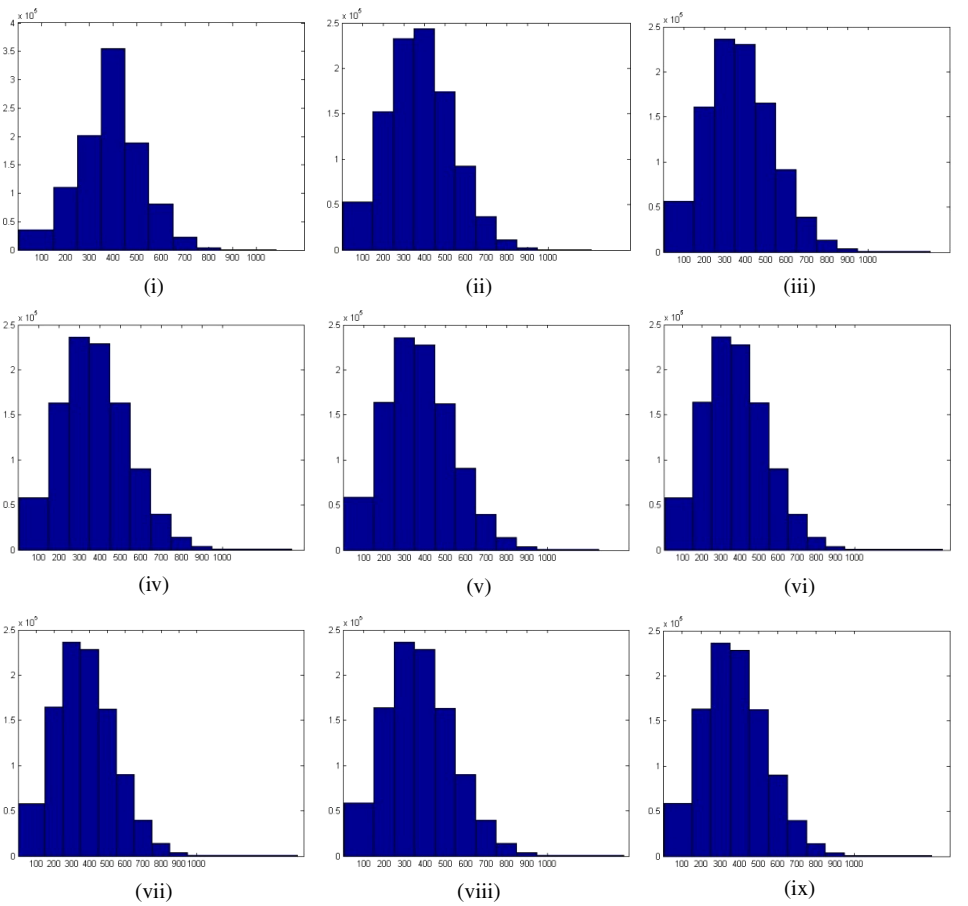

Fig. 6. The calculated velocity distribution with (i) chain length = 10,000, (ii) chain length = 20,000, (iii) chain length = 30,000, (iv) chain length = 40,000, (v) chain length = 50,000, (vi) chain length = 60,000, (vii) chain length = 70,000, (viii) chain length = 80,000 and (ix) chain length = 90,000.

How does $F(t, x, v)$ from DSMC3 (Algorithm 3) change along with the chain length? To answer this question, we record the calculated velocity distribution with different number of chains, and the results are also plotted in Fig. 8.

Figure 8(i) shows the situation when DSMC3 with the 10 chains does not reach the stationary distribution, and the shape of the velocity distribution is far from the Maxwell–Boltzmann distribution. After the number of chains is larger than 10, DSMC3 (Algorithm 3) obtains the stationary distribution, and the shapes of the velocity distribution in Figs. 8(ii) to 8(ix) are almost identical.

4. Conclusions

In this paper, by MCMC and multicore vectorization, we develop DSMC3 (Algorithm 3): a fast solver to calculate the numerical solution for the Boltzmann equation. Computational results show that, DSMC3 (Algorithm 3) is significantly faster than the conventional method DSMC (Algorithm 1). Computational results also show that, the results from DSMC3 (Algorithm 3) is as accurate as the results from the conventional method DSMC (Algorithm 1).

5. Discussion

In this paper, we developed DSMC3 (Algorithm 3), which takes full advantages of multicore of modern CPU. From computational experiments, DSMC3 (Algorithm 3) shows the obvious advantages than the conventional DSMC (Algorithm 1) and the sequential DSMC2 (Algorithm 2), especially the number of chains in DSMC3 (Algorithm 3) is large.

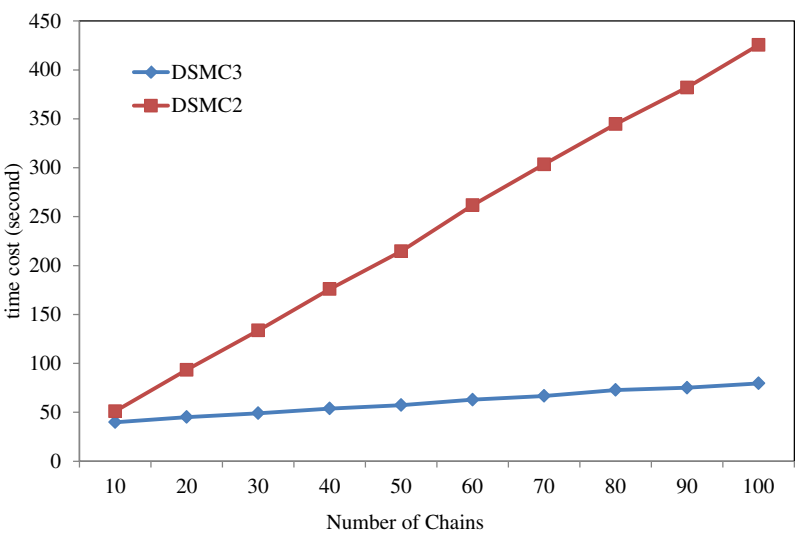

Fig. 7. The time cost of DSMC3 (Algorithm 3) and DSMC2 (Algorithm 2) along with the number of chains.

In current implementation of DSMC3 (Algorithm 3) in this paper, every chain contain single proposal, which means only one possibility is considered in every state. Since multiple proposals in every chain may result in higher efficiency of DSMC3 (Algorithm 3), which means multiple possibility can be considered in every state, we may develop multi-proposal DSMC3 in the future research.

In current implementation of DSMC3 (Algorithm 3) in this paper, the multiple chains in cells are independent, there is no information exchange among chains. However, since few chains may be stuck, and cross-cell information exchange may overcome this problem, we may develop interactional DSMC3 in the future research.

In this paper, DSMC3 (Algorithm 3) is only parallelized by multicore vectorization. Since the goal of parallelization is to take fully advantages of CPU's multicore,

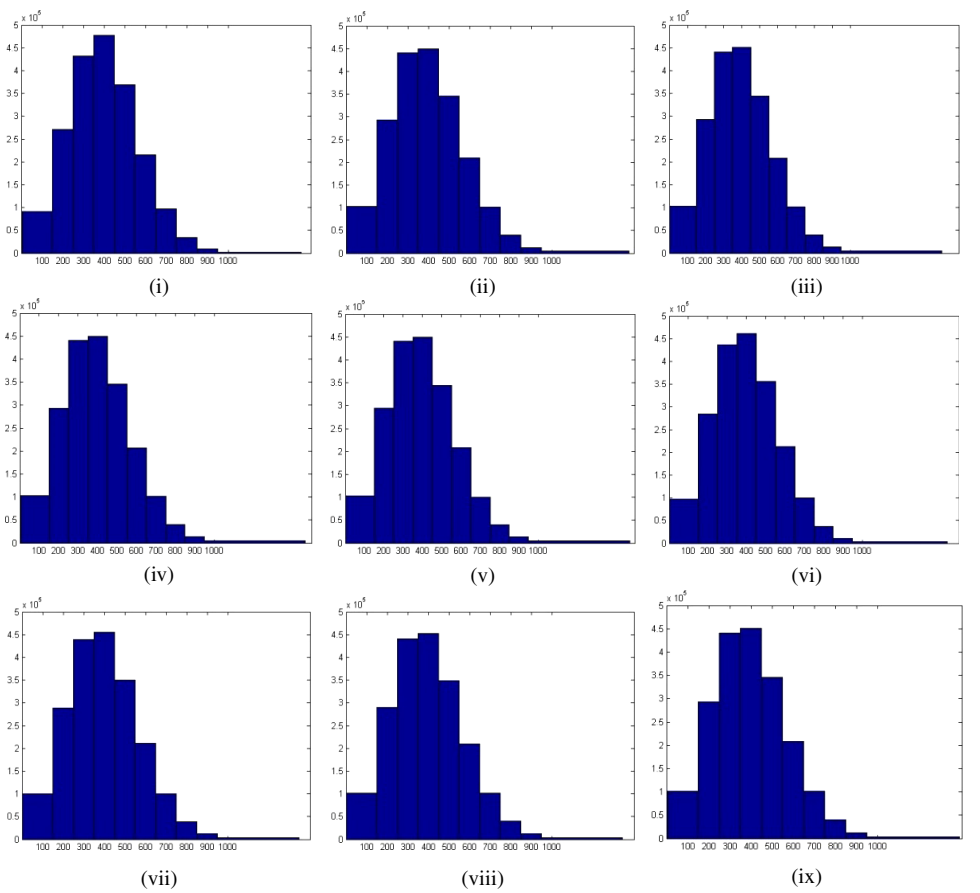

Fig. 8. The calculated velocity distribution: (i) number of chains = 10, (ii) number of chains = 20, (iii) number of chains = 30, (iv) number of chains = 40, (v) number of chains = 50, (vi) number of chains = 60, (vii) number of chains = 70, (viii) number of chains = 80 and (ix) number of chains = 90.

more parallelization techniques such as OpenMP and GPU computing can also be applied, we may develop the OpenMP version, the MPI version of DSMC3 or the GPU version of DSMC3 in the future research.

In this paper, DSMC3 (Algorithm 3) is implemented by MATLAB, and we are working on the C++ implementation of DSMC3 (Algorithm 3) on the platform such as OpenFOAM. By multi-core parallelization and GPU computing, we may distribute DSMC3foam, the OpenFOAM version of DSMC3 (Algorithm 3) in the future research.

References

1. Bird G. A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Clarendon Press, UK, 1994.
2. Zhao D., Acceleration of Hermitian/Skew-Hermitian Separation for Incompressible Navier-Stokes Equation by Parallel Gram-Schmidt Process of GMRES(m), *Fifth M.I.T. Conf. Computational Fluid and Solid Mechanics - Focus*: *Advances in CFD*, Boston, MA, 2009.
3. Zhao D., Preconditioning GMRES(m) for Finite Volume Discretization of Incompressible Navier-Stokes Equation by Hermitian/Skew-Hermitian Separation, *SIAM Conf. Applied Linear Algebra 2009*, Seaside, CA, 2009.
4. Zhao D., HSS Preconditioner for Incompressible Navier-Stokes Equation, *9th Int. Workshop Multiscale (Un)-structured Mesh*, *MIT*, Boston, MA, 2010.
5. Lécot C., A direct simulation Monte Carlo scheme and uniformly distributed sequences for solving the Boltzmann equation, *Computing* **41**(1–2):41–57, 1989.
6. Chonglin Z., Thomas E. S., Consistent Implementation of State-to-State Collision Models for Direct Simulation Monte Carlo, *52nd Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, National Harbor, Maryland, USA, 2014.
7. Kannenberg K. C., Boyd I. D., Strategies for Efficient Particle Resolution in the Direct Simulation Monte Carlo Method, *J. Comput. Phys.* **157**(2):727–745, 2000.
8. Koura K., A sensitive test for accuracy in evaluation of molecular collision number in the direct-simulation Monte Carlo method, *Phys. Fluids A Fluid Dyn.* **2**(7):1287–1289, 1990.
9. Palaniswaamy G., Loyalka., Direct simulation Monte Carlo aerosol dynamics: Collisional sampling algorithms, *Ann. Nucl. Energy* **34**(1–2):13–21, 2007.
10. Zhang C., Schwartzentruber T. E., Inelastic collision selection procedures for direct simulation Monte Carlo calculations of gas mixtures, *Phys. Fluids* **25**, 106105, 1–14, 2013.
11. Wagner W., A convergence proof for Bird's direct simulation Monte Carlo method for the Boltzmann equation, *J. Stat. Phys.* **66**(3–4):1011–1044, 1992.
12. Chen G., Boyd I. D., Statistical Error Analysis for the Direct Simulation Monte Carlo Technique, *J. Comput. Phys.* **126**(2):434–448, 1996.
13. Singh N., Gavasane A., Agrawal A., Analytical solution of plane Couette flow in the transition regime and comparison with Direct Simulation Monte Carlo data, *Comput. Fluids* **97**(0):177–187, 2014.
14. Weinberg M. D., Direct Simulation Monte Carlo for astrophysical flows – I. Motivation and methodology, *Mon. Not. R. Astron. Soc.* **438**(4):1–13, 2014.
15. Weinberg M. D., Direct Simulation Monte Carlo for astrophysical flows – II. Ram-pressure dynamics, *Mon. Not. R. Astron. Soc.* **438**(4):3007–3023, 2014.

16. Chao L., Kwak S. K., Ansumali S., Direct Simulation Monte Carlo for Dense Hard Spheres, *Int. J. Mod. Phys. C* **25**(1):1340023, 2014.
17. Wei J.-L. *et al.*, Modeling the gas flow in the neutralizer of ITER neutral beam injector using Direct Simulation Monte Carlo approach, *Fusion Eng. Des.* **88**(1):46–50, 2013.
18. Wang M., Li Z., Simulations for gas flows in microgeometries using the direct simulation Monte Carlo method, *Int. J. Heat Fluid Flow* **25**(6):975–985, 2004.
19. Economou D. J. *et al.*, Two-dimensional direct simulation Monte Carlo (DSMC) of reactive neutral and ion flow in a high density plasma reactor, *IEEE Trans. Plasma Sci.* **23**(4):581–590, 1995.
20. Sun H., Faghri M., Effect of Surface Roughness on Nitrogen Flow in a Microchannel Using the Direct Simulation Monte Carlo method, *Numer. Heat Transf. A, Appl.* **43**(1):1–8, 2003.
21. Huang W., Bogy D. B., Garcia A. L., Three-dimensional direct simulation Monte Carlo method for slider air bearings, *Phys. Fluids* **9**(6):1764–1769, 1997.
22. Wu J. S., Tseng K. C., Analysis of micro-scale gas flows with pressure boundaries using direct simulation Monte Carlo method, *Comput. Fluids* **30**(6):711–735, 2001.
23. Xue H., Fan Q., Shu C., Prediction of micro-channel flows using direct simulation Monte Carlo, *Probab. Eng. Mech.* **15**(2):213–219, 2000.
24. Coronell D. G., Jensen K. F., Analysis of Transition Regime Flows in Low Pressure Chemical Vapor Deposition Reactors Using the Direct Simulation Monte Carlo Method, *J. Electrochem. Soc.* **139**(8):2264–2273, 1992.
25. Nanbu K., Morimoto T., Suetani M., Direct simulation Monte Carlo analysis of flows and etch rate in an inductively coupled plasma reactor, *IEEE Trans. Plasma Sci.* **27**(5):1379–1388, 1999.
26. Chun J., Koch D. L., A direct simulation Monte Carlo method for rarefied gas flows in the limit of small Mach number, *Phys. Fluids* **17**:107107, 2005.
27. Lempert W. R. *et al.*, Comparison of molecular tagging velocimetry data and direct simulation Monte Carlo simulations in supersonic micro jet flows, *Exp. Fluids* **34**(3):403–411, 2003.
28. Bird G. A. *et al.*, Accuracy and efficiency of the sophisticated direct simulation Monte Carlo algorithm for simulating noncontinuum gas flows, *Phys. Fluids* **21**(1):017103-1–017103-12, 2009.
29. Shuyan W. *et al.*, Flow behavior of clusters in a riser simulated by direct simulation Monte Carlo method, *J. Chem. Eng.* **106**(3):197–211, 2005.
30. Pullin D. I., Direct simulation methods for compressible inviscid ideal-gas flow, *J. Comput. Phys.* **34**(2):231–244, 1980.
31. Roesle M. *et al.*, Analysis of Conduction Heat Loss From a Parabolic Trough Solar Receiver with Active Vacuum by Direct Simulation Monte Carlo, *Numer. Heat Transf. A Appl.* **62**(5):432–444, 2012.
32. Gao D., Schwartzentruber T. E., Optimizations and OpenMP implementation for the direct simulation Monte Carlo method, *Comput. Fluids* **42**(1):73–81, 2011.
33. LeBeau G. J., A parallel implementation of the direct simulation Monte Carlo method. *Comput. Meth. Appl. Mech. Eng.* **174**(3–4):319–337, 1999.
34. Dietrich S., Boyd I. D., Scalar and Parallel Optimized Implementation of the Direct Simulation Monte Carlo Method, *J. Comput. Phys.* **126**(2):328–342, 1996.
35. Wu J. S., Lian Y. Y., Parallel three-dimensional direct simulation Monte Carlo method and its applications, *Comput. Fluids* **32**(8):1133–1160, 2003.
36. Moon B., Saltz J., Adaptive runtime support for direct simulation Monte Carlo methods on distributed memory architectures, *IEEE Scalable High Performance Computing Conf.*, Knoxville, TN, 1994.

37. Goldsworthy M. J., A GPU–CUDA based direct simulation Monte Carlo algorithm for real gas flows, *Comput. Fluids* **94**:58–68, 2014.
38. Zhao D., Ni S., Parallel Multi-Proposal and Multi-Chain MCMC for Calculating *P*-Value of Genome-Wide Association Study, *Parallel Process. Lett.* **23**(3):1350008, 2013.
39. Yu Y., Li K., Tan D., Parallel computation of Entropic Lattice Boltzmann method on hybrid CPU–GPU accelerated system, *Comput. Fluids* **111**:114–121, 2015.
40. Liang J., Li K., Accelerating Solidification Process Simulation for Large-sized System of Liquid Metal Atoms using GPU with CUDA, *J. Comput. Phys.* **257**:521–535, 2014.
41. Li K., Entropic Lattice Boltzmann Method based high Reynolds number flow simulation using CUDA on GPU, *Comput. Fluids* **88**:241–249, 2013.
42. Bird G. A., Approach to Translational Equilibrium in a Rigid Sphere Gas, *Phys. Fluids* **6**(10):1518–1519, 1963.