dugksFoam: An open source OpenFOAM solver for the Boltzmann model equation

Lianhua Zhu[a], Songze Chen[a,*], Zhaoli Guo[a,*]

[a]*State Key Laboratory of Coal Combustion, Huazhong University of Science and Technology, Wuhan, 430074, China*

Abstract

A deterministic Boltzmann model equation solver called dugksFoam has been developed in the framework of the open source CFD toolbox OpenFOAM. The solver adopts the discrete unified gas kinetic scheme (Guo et al., Phys. Rev. E, 91, 033313 (2015)) with the Shakhov collision model. It has been validated by simulating several test cases covering different flow regimes including the one dimensional shock tube problem, a two dimensional thermal induced flow and the three dimensional lid-driven cavity flow. The solver features a parallel computing ability based on the velocity space decomposition, which is different from the physical space decomposition based approach provided by the OpenFOAM framework. The two decomposition approaches have been compared in both two and three dimensional cases. The parallel performance improves significantly using the newly implemented approach. A speed up by two orders of magnitudes has been observed using 256 cores on a small cluster.

Keywords: Boltzmann model equation, OpenFOAM, discrete velocity method, parallel computing

PROGRAM SUMMARY

Program Title: dugksFoam
Journal Reference: CPC Program Library, Queen's University, Belfast, N. Ireland
Catalogue identifier:
Licensing provisions: The MIT License
Programming language: C++
Computer: All capable of running Linux
Operating system: Linux.
RAM: Hundreds of MB to hundreds of GB depending on problem size
Number of processors used: Up to several hundred.
Supplementary material:
Keywords: Boltzmann model equation, Discrete Velocity Method, OpenFOAM, par-

*Corresponding author

Email addresses: `lhzhu@hust.edu.cn` (Lianhua Zhu), `jacksongze@hust.edu.cn` (Songze Chen), `zlguo@hust.edu.cn` (Zhaoli Guo)

allel computing etc.
Classification:
External routines/libraries: OpenFOAM (`http://www.openfoam.org`)
Nature of problem: Solving the Boltzmann equation with Shakhov model explicitly.
Solution method: Discrete unified gas kinetic scheme (DUGKS)
Restrictions: Symmetric boundary condition can only be applied at walls parallel to axis directions.
Running time: Hours to days depending on problem sizes.

1. Introduction

In rarefied gas flow or micro gas flow, the mean free path of molecules λ is comparable to or even larger than the geometric length scale L. At such condition, the velocity distribution of the molecules deviates from the local equilibrium state due to insufficient inter-molecular collisions. The well-known Euler equations or Navier-Stokes-Fourier (NSF) equations are not accurate in describing such non-equilibrium flows [1]. A common criterion for the breakdown of the NSF equation is $\mathrm{Kn} < 0.001$, where $\mathrm{Kn} = \lambda/L$ is the Knudsen number. For $0.001 < \mathrm{Kn} < 0.1$, the NSF equation can still give reasonable results provided that the wall boundaries are treated with slip boundary condition. For even larger Knudsen number ($\mathrm{Kn} > 0.1$), the Boltzmann equation should be used as the governing equation [1].

The Boltzmann equation is a classical kinetic equation and reads as follows,

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f = \Omega(f, f) \tag{1}$$

where $f = f(\boldsymbol{v}, \boldsymbol{x}, t)$ is the velocity distribution function of particles with velocity $\boldsymbol{v}$ at $\boldsymbol{x}$ and time t. The RHS of the equation is the collision term which involves a complex integral expression. The high-dimensionality of the Boltzmann equation make it extremely expensive to solve directly in the discrete phase space in a deterministic way. Early attempts to solve the Boltzmann equation are limited to two-dimensional or axisymmetric flows [2, 3, 4, 5, 6, 7, 8, 9]. Up to now, the workhorse for practical high-speed non-equilibrium flows is the direct simulation Monte Carlo (DSMC) method [1] which is a stochastic approach for the Boltzmann equation. However, with the fast growing of computational power and the increasing demand of modeling low-speed micro flows in emerging Microelectromechanical Systems (MEMS) industry, there is a renewed interest in deterministic methods for Boltzmann equation or its model equations [10], because the deterministic approach has many advantages over the DSMC method. For example, some asymptotic preserving schemes have been developed for multiscale simulations [11, 12] and implicit schemes for accelerating steady flows simulations [4, 13, 14]. Another advantage of the deterministic approach is that the dimensionality of phase space can be reduced for low dimensional problems [15] when adopting the relaxation time approximated collision models [16, 17, 18].

Even though the processing speed of single central processing unit (CPU) core has improved significantly in the last decades, practical solving of the Boltzmann equation or even its model equations in three-dimensional space still needs parallel computation, considering the computational complexity and the large memory consumption. There have been several parallel three-dimensional deterministic Boltzmann solvers reported in the literature [19, 20, 21, 22, 23, 24, 25]. But few open-source deterministic Boltzmann model equation solvers are available with unstructured mesh ability. On the other hand, many open-source parallel DSMC codes are available, such as the dsmcFoam [26]. In this paper, we will present an open-source deterministic Boltzmann model equation solver entitled *dugksFoam*. The solver is developed in the framework of OpenFOAM, which is a popular high-level toolbox for computational continuum mechanics (mainly for computational fluid dynamics) [27, 28]. The kinetic model used is the Shakhov model [18], and the numerical scheme employed is the recently proposed discrete unified gas kinetic schemes (DUGKS) [29], which is a discrete-velocity method with asymptotic preserving property. The DUGKS has been extended to unstructured mesh previously in the framework of OpenFOAM as a prototype of this solver [30].

The rest of this paper is organized as following. In Sec. 2, the Boltzmann-Shakhov model and the DUGKS are presented. In Sec. 3, we explain the implementation of dugksFoam in the framework of OpenFOAM and discuss two different domain decomposition strategies for message-passing-interface (MPI) communication, i.e., physical space decomposition and velocity space decomposition. In Sec. 4, a series of benchmark tests covering both low-speed and high-speed flows in a wide range of Knudsen numbers are simulated to validate this solver. In Sec. 5, the parallel performance is evaluated in detail by simulating both two-dimensional and three-dimensional flows. In the last section, some comments are made.

2. Theoretical background and numerical scheme

2.1. The Boltzmann-Shakhov model

The Shakhov model is a relaxation time approximation of the original Boltzmann collision kernel. Unlike the well-known BGK approximation, in Shakhov model the Prandtl number can be adjusted freely [18]. The Boltzmann equation with Shakhov model in D dimensional spatial space reads as,

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = -\frac{1}{\tau}\left[f - f^S\right], \tag{2}$$

where $f = f(\boldsymbol{\xi}, \boldsymbol{\eta}, \boldsymbol{\zeta}, \boldsymbol{x}, t)$ is the velocity distribution function of particles with velocity $\boldsymbol{v} = (\boldsymbol{\xi}, \boldsymbol{\eta}) = (\xi_1, \ldots, \xi_D, \eta_{D+1}, \ldots, \eta_3)$ in three dimensional velocity space at position $\boldsymbol{x} = (x_1, \ldots, x_D)$ and time t. For example, for one-dimensional problems, $D = 1$, $\boldsymbol{\xi} = (v_1)$ and $\boldsymbol{\eta} = (v_2, v_3)$, while for two-dimensional problems, $D = 2$, $\boldsymbol{\xi} = (v_1, v_2)$ and $\boldsymbol{\eta} = (v_3)$. The partitioning of $\boldsymbol{v}$ into $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ is a preparation for dimension reduction. $\boldsymbol{\zeta}$ is vector of length K representing the internal degree of freedom. f^S is the Shakhov equilibrium distribution function given by the Maxwellian distribution function f^M plus a heat flux correction term

$$\begin{aligned} f^S &= f^M \left[1 + (1 - \mathrm{Pr})\frac{\boldsymbol{c}\cdot\boldsymbol{q}}{5pRT}\left(\frac{c^2+\eta^2}{RT} - 5\right)\right] = f^M + f_{\mathrm{Pr}}, \\ f^M &= \frac{\rho}{(2\pi RT)^{(3+K)/2}} \exp\left(-\frac{c^2+\eta^2+\zeta^2}{2RT}\right), \end{aligned} \tag{3}$$

where Pr is the Prandtl number and $\boldsymbol{c} = \boldsymbol{\xi} - \boldsymbol{U}$ is the peculiar velocity around the fluid velocity $\boldsymbol{U}$; ρ, T, $\boldsymbol{q}$ are the density, temperature and heat flux, respectively. R is the specific gas constant. The pressure p is related to the density and temperature by $p = \rho RT$. The collision time τ in Eq. (2) is calculated from the dynamic viscosity μ and the pressure p by $\tau = \mu/p$. The dynamic viscosity μ depends on the temperature as

$$\mu = \mu_{\mathrm{ref}}\left(\frac{T}{T_{\mathrm{ref}}}\right)^{\omega}, \tag{4}$$

where μ_{ref} is the viscosity at the reference temperature T_{ref}, and the exponent ω is a constant depends on the inter-molecular interaction model.

The conservative flow variables $\boldsymbol{W} \equiv (\rho, \rho\boldsymbol{U}, \rho E)^T$ are calculated as moments of the distribution function,

$$\boldsymbol{W} = \int \boldsymbol{\psi} f \mathrm{d}\boldsymbol{\xi}\mathrm{d}\boldsymbol{\eta}\mathrm{d}\boldsymbol{\zeta}, \tag{5}$$

where $\boldsymbol{\psi} = \left(1, \boldsymbol{\xi}, \frac{1}{2}(\xi^2+\eta^2+\zeta^2)\right)^T$ and $\rho E = \frac{1}{2}\rho U^2 + C_{\mathrm{V}}T = \frac{1}{2}\rho U^2 + p/(\gamma-1)$, with $C_{\mathrm{V}} = (3+K)\rho R/2$ and $\gamma = (K+5)/(K+3)$ being the heat capacity at constant volume and the specific heat ratio, respectively. The heat flux $\boldsymbol{q}$ is defined by

$$\boldsymbol{q} = \frac{1}{2}\int \boldsymbol{c}(c^2+\eta^2+\zeta^2) f \mathrm{d}\boldsymbol{\xi}\mathrm{d}\boldsymbol{\eta}\mathrm{d}\boldsymbol{\zeta}. \tag{6}$$

The dependencies of f on $\boldsymbol{\zeta}$ for rotational-equilibrium flows and on $\boldsymbol{\eta}$ for lower dimensional flows ($D < 3$) can be reduced using the standard procedure proposed by Chu [15]. By introducing the flowing velocity distribution functions,

$$\Phi = \left[\begin{array}{c} g \\ h \end{array}\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[\begin{array}{c} 1 \\ \eta^2+\zeta^2 \end{array}\right] f(\boldsymbol{\xi}, \boldsymbol{\eta}, \boldsymbol{\zeta}, \boldsymbol{x}, t)\mathrm{d}\boldsymbol{\eta}\mathrm{d}\boldsymbol{\zeta} \tag{7}$$

and integrating Eq. (2) over the range of $[-\infty, \infty]$ with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\zeta}$, we have

$$\frac{\partial\Phi}{\partial t} + \boldsymbol{\xi}\cdot\boldsymbol{\nabla}\Phi = -\frac{1}{\tau}[\Phi - \Phi^S], \tag{8}$$

where the reduced equilibrium distribution functions g^S and h^S are

$$\begin{aligned}
g^S &= g^M \left[1 + (1-\mathrm{Pr})\frac{\boldsymbol{c}\cdot\boldsymbol{q}}{5pRT}\left(\frac{c^2}{RT} - D - 2\right)\right], \\
h^S &= g^M \left\{K + 3 - D + (1-\mathrm{Pr})\frac{\boldsymbol{c}\cdot\boldsymbol{q}}{5pRT}\left[\left(\frac{c^2}{RT} - D\right)(K+3-D) - 2K\right]\right\} RT, \\
g^M &= \frac{\rho}{(2\pi RT)^{D/2}} \exp\left[-\frac{c^2}{2RT}\right].
\end{aligned} \tag{9}$$

The conservative macroscopic variables can be computed from these reduced distribution functions as

$$\rho = \int g \mathrm{d}\boldsymbol{\xi}, \quad \rho\boldsymbol{U} = \int \boldsymbol{\xi} g \mathrm{d}\boldsymbol{\xi}, \quad \rho E = \frac{1}{2}\int (\xi^2 g + h)\mathrm{d}\boldsymbol{\xi}, \tag{10}$$

and the heat flux can be computed as

$$\boldsymbol{q} = \frac{1}{2}\int \boldsymbol{c}(c^2 g + h)\mathrm{d}\boldsymbol{\xi}. \tag{11}$$

2.2. Discrete unified gas kinetic scheme

The discrete unified gas kinetic scheme (DUGKS) is a finite-volume scheme for the discrete-velocity Boltzmann model [29]. The governing equation is firstly discretized in the velocity space with chosen discrete velocity points $\{\boldsymbol{\xi}_\alpha, \alpha = 1, 2, ..., M\}$,

$$\frac{\partial \Phi_\alpha}{\partial t} + \boldsymbol{\xi}_\alpha \cdot \boldsymbol{\nabla}\Phi_\alpha = -\frac{1}{\tau}[\Phi_\alpha - \Phi_\alpha^S] \equiv \Omega_\alpha, \tag{12}$$

where Φ_α and Φ_α^S are the distribution function and equilibrium distribution function with discrete velocity $\boldsymbol{\xi}_\alpha$. Equation (12) is then discretized in the spatial space with the following cell-centered finite-volume scheme [29],

$$\Phi_{\alpha,k}^{n+1} - \Phi_{\alpha,k}^{n} + \frac{\Delta t}{|V_k|}\mathcal{F}_{\alpha,k}^{n+1/2} = \frac{\Delta t}{2}[\Omega_{\alpha,k}^{n+1} + \Omega_{\alpha,k}^{n}], \quad k = 1, 2, \ldots, N, \tag{13}$$

where $\Phi_{\alpha,k}^n$ is the cell averaged value of Φ_α in cell k at time level t^n, $|V_k|$ is the volume of the cell, N is total number of cells and $\Delta t = t^{n+1} - t^n$ is the time step. The flux $\mathcal{F}_{\alpha,k}^{n+1/2}$ is evaluated at middle time step by [30],

$$\mathcal{F}_{\alpha,k}^{n+1/2} = \sum_l \boldsymbol{\xi}_\alpha \cdot \boldsymbol{S}_{k,l}\Phi_{\alpha,k,l}^{n+1/2}, \tag{14}$$

where $\boldsymbol{S}_{k,l}$ is the surface vector of face l belonging to cell k, and $\Phi_{\alpha,k,l}^{n+1/2}$ is the distribution function at the center of face l at the middle time step. In DUGKS, the distribution functions at cell faces are constructed in a physical way by solving the governing equation locally along the characteristic line that ends at the cell face center $\boldsymbol{x}_f$ from t^n to $t^{n+1/2}$,

$$\Phi_\alpha^{n+1/2}(\boldsymbol{x}_f) - \Phi_\alpha^n(\boldsymbol{x}_f - \boldsymbol{\xi}_\alpha s) = \Delta s/2 \left[\Omega_\alpha^{n+1/2}(\boldsymbol{x}_f) + \Omega_\alpha^n(\boldsymbol{x}_f - \boldsymbol{\xi}_\alpha s)\right], \tag{15}$$

where $s = t^{n+1/2} - t^n$ is the half time step. Equation (15) can be rewritten in an explicit form by introducing $\bar{\Phi} = \Phi - s/2\Omega$ and $\bar{\Phi}^+ = \Phi + s/2\Omega$,

$$\bar{\Phi}_\alpha^{n+1/2}(\boldsymbol{x}_f) = \bar{\Phi}_\alpha^{+,n}(\boldsymbol{x}_f - \boldsymbol{\xi}_\alpha \Delta t/2). \tag{16}$$

$\bar{\Phi}_\alpha^{+,n}(\boldsymbol{x}_f - \boldsymbol{\xi}_\alpha \Delta t/2)$ is calculated using first order Taylor expansion from the upstream cell center [30]. The gradients of $\bar{\Phi}_\alpha^{+,n}$ at cell centers are evaluated using Gauss linear scheme or least square method [31]. The accuracy of gradient evaluation using Gauss linear scheme and the least square method is similar for simple regular mesh of quadrilateral or hexahedral elements [32]. Generally, the least square approach is more tolerant to mesh distortions which are inevitable in unstructured meshes of complex geometries [33], but for highly stretched meshes in the presence of curvature, the Gauss linear scheme is more accurate [34]. The calculated gradients are limited with the Venkatakrishnan limiter which is a popular choice for unstructured meshes [31]. The strength of the limiter can be adjusted by a factor in the range of $[0, 1]$, where 0 means no limiting and 1 means full limiting [28]. After getting $\bar{\Phi}^{n+1/2}(\boldsymbol{x}_f)$, the macro variables $W^{n+1/2}(\boldsymbol{x}_f)$ can be obtained by taking moments of $\bar{\Phi}^{n+1/2}(\boldsymbol{x}_f)$ due to the compatibility condition. Then the original distribution function $\Phi^{n+1/2}(\boldsymbol{x}_f)$ can be recovered from the definition of $\bar{\Phi}$.

Equation (13) can be rewritten in the following explicit form by introducing another two transformed distribution functions, $\tilde{\Phi} = \Phi - \Delta t/2\Omega$ and $\tilde{\Phi}^+ = \Phi + \Delta t/2\Omega$,

$$\tilde{\Phi}_{\alpha,k}^{n+1} = \tilde{\Phi}_{\alpha,k}^{+,n} - \frac{\Delta t}{|V_k|}\mathcal{F}_{\alpha,k}^{n+1/2}. \tag{17}$$

In the actual implementation, $\tilde{\Phi}$ is tracked instead of Φ. $\tilde{\Phi}^+$ and $\bar{\Phi}^+$ are calculated from $\tilde{\Phi}$ by [29]

$$\bar{\Phi}^+ = \frac{2\tau - s}{2\tau + \Delta t}\tilde{\Phi} + \frac{3s}{2\tau + \Delta t}\Phi^S, \quad \tilde{\Phi}^+ = \frac{2\tau - \Delta t}{2\tau + \Delta t}\tilde{\Phi} + \frac{2\Delta t}{2\tau + \Delta t}\Phi^S. \tag{18}$$

The time step in the DUGKS is determined by the Courant-Friedrichs-Lewy (CFL) condition,

$$\Delta t = \alpha \left(\frac{\Delta x}{|\boldsymbol{U}| + |\boldsymbol{\xi}|}\right)_{\min}, \tag{19}$$

where $0 < \alpha < 1$ is the CFL number and Δx is the distance between the centers of two adjacent cells that share an interface. Appropriate velocity grid is chosen according to a prior estimation of the deviation of the distribution function from the equilibrium distribution. The moments (macro variables) are approximated using numerical quadrature. For low-speed near-equilibrium flows, the Gauss-Hermit or half-range Gauss-Hermit quadrature are common choices. While for high-speed or highly non-equilibrium flows, composite Newton-Cotes quadrature is more appropriate.

The major evolving steps of $\tilde{\Phi}$ and $\boldsymbol{W}$ in DUGKS are listed below [29],

1. Initialize $\tilde{\Phi}_{\alpha}^{0}$ using the equilibrium distribution in each cell center;
2. Calculate $\bar{\Phi}_{\alpha}^{+,n}$ and their limited gradients at cell centers;
3. Calculate $\bar{\Phi}_{\alpha}^{n+1/2}$ at cell faces;
4. Calculate $\boldsymbol{W}^{n+1/2}$ at cell faces;
5. Calculate $\Phi^{n+1/2}$ at cell faces;
6. Calculate $\tilde{\Phi}^{n+1}$ at cell centers;
7. Calculate $\boldsymbol{W}^{n+1}$ at cell centers;
8. If not converged, go back to Step 2.

Because DUGKS is an explicit scheme, for steady problems, the flow field will be assumed to be steady when the average relative change of the macro fields in two-successive steps are less than a given tolerance ϵ (10^{-8} for instance),

$$\varepsilon^n = \frac{\sum_i |\mathcal{W}^{n+1} - \mathcal{W}^n|}{\sum_i \mathcal{W}_i^n} < \epsilon, \quad \text{for} \quad \mathcal{W} \in \{\rho, \boldsymbol{U}, T\}, \tag{20}$$

where the summations are taken over all cells.

3. Implementation in OpenFOAM

The OpenFOAM is essentially a numerical solving environment for partial differential equations (PDE) commonly seen in fluid dynamics. It is developed using the C++ programming language and applies the object-oriented programming (OOP) and generic programming techniques intensively. The variables to be solved in the PDEs are abstracted as geometrical fields (`GeometricField`) [35], which is essentially an encapsulation of the discrete field data associated with the geometrical mesh and its boundary information. Common mathematical tensor operators such as `+`,`-`, scalar product and vector product have been reloaded or implemented at the field level to simplify the calculations of `GeometricField`. The OpenFOAM also provides many differential operators such as the gradient operator and the Laplacian operator to manipulate the fields. Moreover, different discretization schemes for the differential operators can be chosen or even be implemented by the users.

Applying the OpenFOAM field operation techniques to the relaxation time approximated Boltzmann model equations is straightforward, because after discretizing of the governing equation in the velocity space, each of the discrete-velocity kinetic equations is a simple linear convection equation with a source term. The distribution functions at each discrete velocity points are defined as scalar type `GeometricField`. The solving procedures are then mapped to a series of C++ expressions of field operations and manipulations.

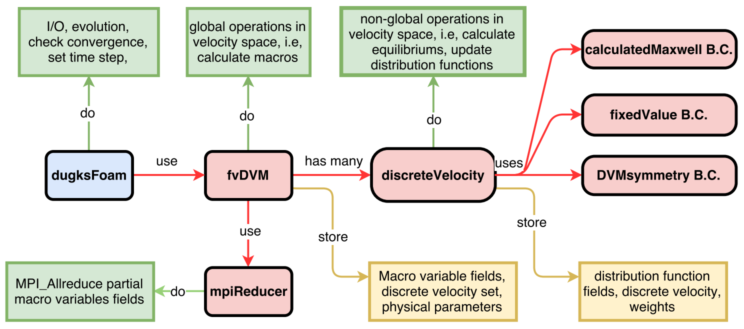

I/O, evolution, check convergence, set time step,
global operations in velocity space, i.e, calculate macros
non-global operations in velocity space, i.e, calculate equilibriums, update distribution functions
calculatedMaxwell B.C.
fixedValue B.C.
DVMsymmetry B.C.
do
do
do
dugksFoam
use
fvDVM
has many
discreteVelocity
uses
store
store
use
MPI_Allreduce partial macro variables fields
do
mpiReducer
Macro variable fields, discrete velocity set, physical parameters
distribution function fields, discrete velocity, weights

Figure 1: Main components of dugksFoam.

3.1. Structure of the solver

In dugksFoam, we organized the data fields and solving procedures into a macroscopic level and a microscopic level. Accordingly, two main classes, i.e., the `fvDVM` and the `discreteVelocity` have been implemented, as illustrated in Fig. 1. The `fvDVM` stores the global information with respect to the discrete velocity space, such as macroscopic fields, physical parameters, solving control parameters and the discrete velocity set. There is only one instance of `fvDVM` in the running solver. The `discreteVelocity`, on the other hand, stores the data specific only to a certain discrete velocity, i.e., the discrete velocity $\boldsymbol{\xi}_\alpha$, its weight w_α and the distribution functions fields $\tilde{\phi}_\alpha$, $\tilde{\phi}^+_\alpha$, $\bar{\phi}_\alpha$, $\bar{\phi}^+_\alpha$ and $\boldsymbol{\nabla}\bar{\phi}^+_\alpha$ appearing in the evolution of DUGKS. The `fvDVM` keeps an array of references to all of the `discreteVelocity` instances, from which it can evaluate the moments and then update the macro fields.

3.2. Implementation of boundary conditions

The distribution functions coming into the computational domain from the boundaries have to be specified and should be consistent with the physical boundary information which are often given in terms of macro variables. We have implemented several commonly used boundary condition (B.C.) types for the distribution function fields. The specifications of those B.C. types at each boundary patch is indicated by the corresponding density field boundary conditions which are provided initially in the file `0/rho` of a standard OpenFOAM run case, because only macro fields are provided in the initialization. The rule of mapping of B.C. types from the density field to the distribution function fields are listed in Table. 1.

In dugksFoam, walls are treated as purely diffusive boundaries, which means the incoming distribution function from the walls are set to be Maxwellian and no-penetration condition is satisfied. For free-stream boundaries, the incoming distribution functions are also set to be Maxwellian and only depends on the free stream macroscopic condition. For symmetric boundaries, specular reflective boundary for the distribution function fields is applied. Besides the B.C. types listed in Table. 1, there are also `cyclic` and `processor` B.C. types

Physical B.C.	Macro density	Distribution functions
wall	`calculatedMaxwell*`	`maxwellWall*`
free streaming	`fixedValue`	`mixed*`
symmetric	`symmetryMod*`	`DVMsymmetry*`

Table 1: Map of boundary condition type between the macro density field and distribution function field. Items without * are boundary condition types provided by OpenFOAM.

which are provided by OpenFOAM ready to be used in dugksFoam. The `cyclic` B.C. type is used for periodical boundaries and `processor` B.C. type is assigned to communication interfaces automatically after the physical domain decomposition (see below).

3.3. MPI Parallel computation

Parallel computing is an indispensable feature for any practical kinetic equation solver due to the high computational cost in terms of both floating point operations and memory consumptions. Because of the huge memory consumptions of kinetic equation solvers, their parallelization have to adopt the distributed memory model which is often popularly implemented using a Beowulf cluster and the message-passing-interface (MPI) library for inter-process communication. In parallel computing, the computation task is divided and assigned to the all of the processors evolved. This process is referred as task decomposition. In Euler/Naiver-Stokes equation based CFD solvers, the task decomposition is simply the partitioning of the physical space mesh. While for direct kinetic equation solver, the high dimensionality of equation and the locality of the collision term offer more flexibility in choosing the task decomposition method [3]. The computation task can be decomposed in either the spatial space or the velocity space. There has been several investigations and comparisons of different decomposition strategies in the literature [3, 19, 21, 14]. For example, Titarev et al. compared the parallel efficiency of the physical space decomposition (PSD) and the velocity space decomposition (VSD) approaches for their three dimensional implicit Boltzmann solver and demonstrated its good scalability over one thousands of processors using the PSD approach [14]. Beside the VSD and PSD approaches, a hybrid decomposition approach was also proposed [25], in which the physical space is decomposed and the communication boundary data are exchanged using MPI, and in each physical sub-domain the discrete velocity space is decomposed and the sub-task is further parallelized using the OpenMP [25]. This approach can take advantages of modern multi-core CPUs and large memory size.

In the current version of dugksFoam, both the PSD and the VSD strategies are implemented. The actual decomposition method can be specified through an option when starting dugksFoam.

3.3.1. Physical space decomposition (PSD)

By the PSD, each CPU core processes only a sub-domain of the whole physical computation domain but accounts for all discrete velocities. The ability of PSD based parallel computation in the dugksFoam is provided by the official OpenFOAM. The OpenFOAM has implemented an elegant way employing the *zero-halo-cell* [36] concept to account for the communication between the sub-domains. The communication boundaries are assigned with the `processor` type B.C., which is just like normal types of B.C.. The boundary field of the `processor` type boundary patch serves as the role of halo-cells data in many other halo-cell based domain decomposition solvers. Using such a design, the MPI communications are transparent to solver developers.

Applying the `processor` type B.C. at communication boundaries of the distribution function fields means that for each of the discrete velocities, a pair of MPI send and receive functions will be called at every communication boundary at each time step. If the number of discrete velocities is large but the number of cells is small, the communication network will be overwhelmed by a large number of tiny messages. In addition, it is found that in OpenFOAM even though the non-blocking version of MPI send/receive can be used, the computation in the bulk internal domain is not overlapped with the communication. Considering the above factors, the parallel efficiency of the current solver is not expected to be high using the PSD approach, particularly for high speed or highly non-equilibrium flows which require a large number of discrete velocities.

3.3.2. Velocity space decomposition (VSD)

By the VSD approach, each CPU core only processes a subset of the discrete velocities but accounts for all of the physical space cells. The communication only occurs when evaluating the moments. Before the communication, each core will hold only a fraction of the moments which are calculated from the discrete velocities belonging to the core. After the collective communication, each core will get the completed moments. The communication can be easily done by calling the `MPI_Allreduce` procedure. Actually, this approach is relatively simpler to implement than the PSD based one. We implement the class `fieldMPIreducer` to do the communication job in dugksFoam (see in Fig. 1).

The VSD, however, also comes with several disadvantages. When performing global reductions, each message contains the moments of the whole physical domain, which means for a large cell numbers, the message sizes is very large. This can results in deficient communications and hence declined parallel efficiency for large scale 3D problems with millions of cells. In addition, global reductions involving a large number of MPI processes also tend to be inefficient because they are natural barriers between computations.

4. Benchmark tests

We use three benchmark tests to verify the implementation of dugksFoam. The first one is the one-dimensional shock tube problem in all flow regimes. The second one is a thermal creep flow problem in a square cavity in slip and transition regimes. The last one is the lid-driven cubic cavity flow in the transition regime. For each of the tests, our results are compared with those in literature or DSMC results. The setups of the three cases have been included in the source code package.

4.1. Shock tube problem

This case is a classical benchmark problem for compressible Euler or NS solvers. In this work, we compute it in different flow regimes. The parameters are set to be identical with those in [29]. For this problem, the parameters are often given in the non-dimensional form. However in OpenFOAM, all input physical parameters and flow fields are defined with dimensions. We still use the nondimensionalized values in the setup as if we are solving a modeled dimensional flow system with a virtual type of gas molecule. The computation domain is $-0.5 \leq x \leq 0.5$ and the initial density, velocity and pressure are set to be

$$(\rho, U, p) = \begin{cases} (\rho_1, U_1, p_1) = (1.0, 0.0, 1.0) & x \leq 0; \\ (\rho_2, U_2, p_2) = (0.125, 0.0, 0.1) & x > 0. \end{cases} \tag{21}$$

The specific gas constant is $R = 0.5$, such that the initial temperature in the left part of the domain is $T_1 = 2$. The gas is modeled as hard-sphere molecules such that the viscosity-temperature dependence is $\mu = \mu_{\mathrm{ref}}(T/T_{\mathrm{ref}})^{0.5}$, where T_0 is the reference temperature. The reference viscosity is related to the reference mean free path λ_0 by [1]

$$\lambda_0 = \frac{16}{5}\frac{\mu_0}{p_0}\sqrt{\frac{RT_0}{2\pi}}, \tag{22}$$

where p_0 is the reference pressure. The left initial state is taken as the reference state. Using the domain length as reference length, the characteristic Knudsen number is $\mathrm{Kn} = \lambda_0$. By adjusting μ_0 from 10^{-5} to 10, Kn varies from 1.277×10^{-5} to 12.77. The internal degree of freedom is $K = 2$, and the Prandtl number is $\mathrm{Pr} = 2/3$. The computational domain is divided into 100 uniform cells and the time step size is fixed at $\Delta t = 0.04$. Such a configuration is the same as Ref. [29]. The strength of the gradient limiter is set to be $\psi = 1$, which means full limiting (Sec. 2.2). The simulations stop at $t_{\mathrm{end}} = 0.15$, at which the flow fields are compared with the results in [29].

The density, temperature and velocity distributions with $\mu_0 = 10$, 0.1 and 10^{-5} are shown in Figs. 2-4 together with the results from Ref. [29]. It can be seen that the results of dugksFoam match with those in Ref. [29] accurately in general. While at $\mu_0 = 10^{-5}$, the overshot of the velocity profile at discontinuities predicted by dugksFoam is slightly larger than that in Ref. [29]. This difference can be explained by the fact that dugksFoam uses the Venkatakrishnan limiter which is different from the van Leer limiter adopted by Ref. [29].

4.2. Thermally induced flow in a square cavity

At micro or rarefied conditions, the temperature inhomogeneity of a gas system can lead to a variety of flow phenomenon [8]. In this test, we consider such a thermally induced flow using the configurations as illustrated in Fig. 5. The length of the square cavity is L and center of the cavity locates at $(0, 0)$.

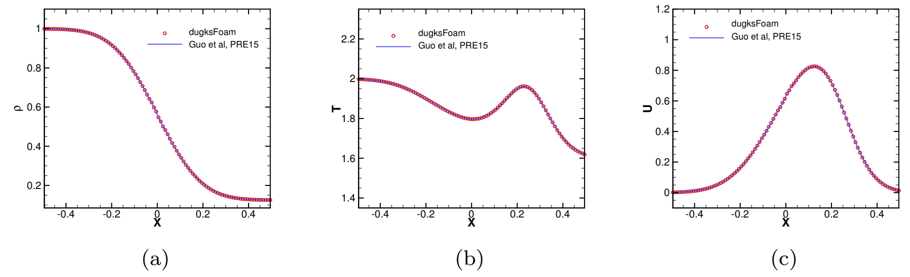


Figure 2: (a) Density, (b) temperature and (c) velocity profiles for the shock tube case at $\mu_{\text{ref}} = 10$ (Kn = 12.77).

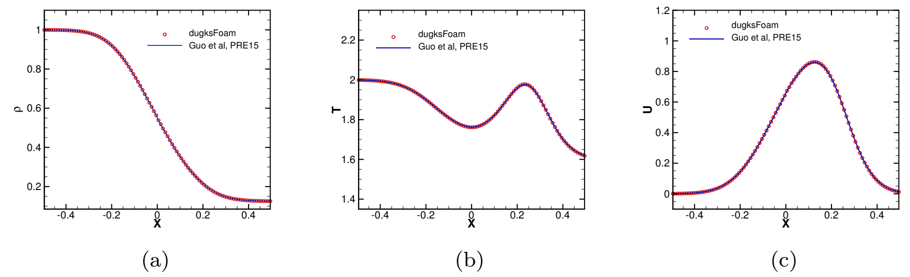


Figure 3: (a) Density, (b) temperature and (c) velocity profiles for the shock tube case at $\mu_{\text{ref}} = 0.1$ (Kn $= 1.277 \times 10^{-1}$).

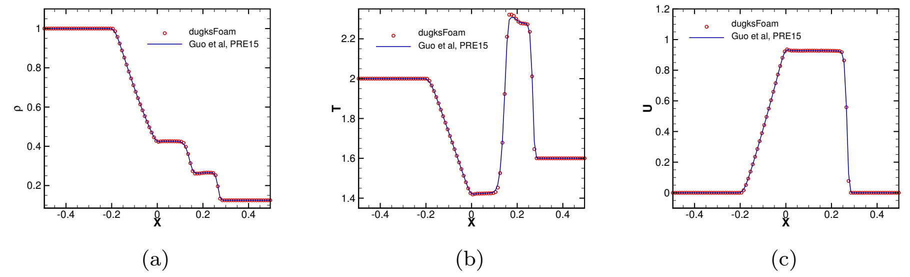


Figure 4: (a) Density, (b) temperature and (c) velocity profiles for the shock tube case at $\mu_{\text{ref}} = 1 \times 10^{-5}$ (Kn $= 1.277 \times 10^{-5}$).

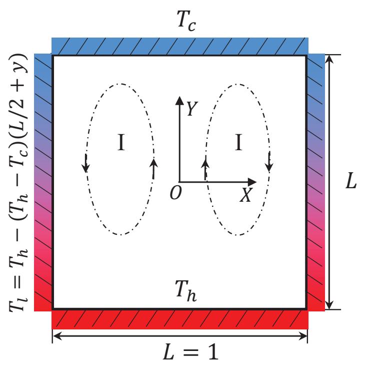
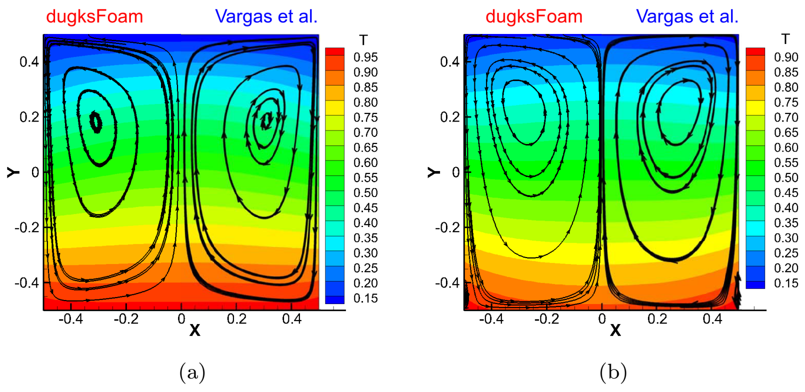


Figure 5: Illustration of the thermally induced flow in a square cavity.



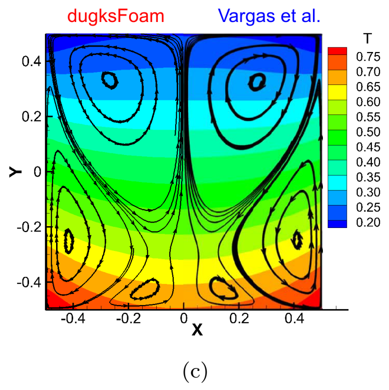
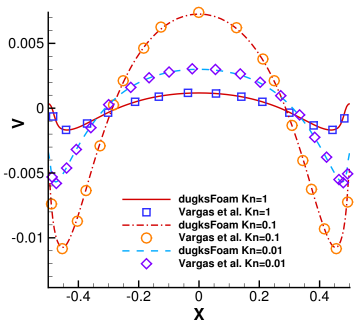
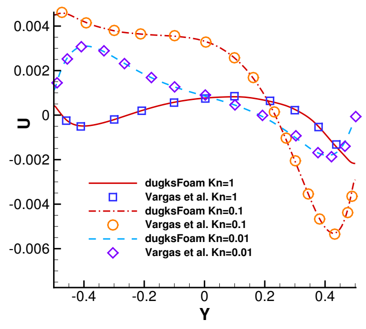


Figure 6: Temperature contours and velocity streamlines at (a) Kn = 0.01, (b) Kn = 0.1 and (c) Kn = 1. In each of the sub-figure, the left half shows the results of dugksFoam, the right half shows the results extracted from Ref. [37].



(a)



(b)

Figure 7: Profiles of the (a) U and (b) V components of the velocity on horizontal and vertical lines, respectively, passing through the centers of the left primary vortex at varies Knudsen numbers for the thermally induced flow case.

The top and bottom wall are kept isothermal with temperature T_c and T_h respectively. The lateral walls are imposed a linear temperature distribution from T_c at the top to T_h at the bottom, i.e., $T_l(y) = T_h - (T_h - T_c)(0.5 + y)$. All of the boundaries are assumed to be purely diffusive walls. In such a configuration, complex flow patterns can develop in the cavity depending on the characteristic Knudsen number. This problem has been investigated recently by Vargas et al. [37] using both the Shakhov kinetic equation and the DSMC method. We use the same parameters in one of their configurations in order to compare our results with theirs. The temperature ratio is $T_h/T_c = 10$, and the gas is modeled as monatomic hard-sphere molecules ($K = 0$, $\omega = 0.5$). The characteristic Knudsen number is defined as [37]

$$\mathrm{Kn}_{\mathrm{ref}} = \frac{\sqrt{\pi}}{2}\frac{\mu_{\mathrm{ref}}\sqrt{2RT_h}}{p_{\mathrm{ref}}L}, \tag{23}$$

where μ_{ref} is the reference viscosity at reference temperature T_h, and p_{ref} is the reference pressure corresponding to the initial uniform density field at T_h. By adjusting μ_{ref} in the setups, different Knudsen numbers can be achieved. In this test, we consider three cases, i.e., Kn = 0.01, 0.1 and 1.

For the physical space, we use a non-uniform mesh with 80×80 rectangular cells. The cell size are graded increasingly towards the cavity center. The smallest cell size is around 0.005×0.005. The CFL number is 0.8. The gradient scheme used is the Gauss linear scheme with full limiting. The velocity space is discretized using 28×28 half-range Gauss-Hermit quadrature points in the cases of Kn = 0.01 and 0.1, and 161×161 uniform points in the range of $[-4\sqrt{RT_h}, 4\sqrt{RT_h}] \times [-4\sqrt{RT_h}, 4\sqrt{RT_h}]$ for the case of Kn = 1. It should be noted that both the number of physical space cells and the number of discrete velocity points are much smaller than those used in [37]. Nevertheless, our results agree with Ref. [37] quite well, as will be shown in the following.

Fig. 6 shows the side-by-side comparisons of the temperature distributions and velocity streamlines predicted by the current solver and the results extracted from Ref.[37]. Excellent agreements between the two results can be observed. In the dugksFoam result of Kn = 0.1, even the tiny second pair of vortexes near the bottom corners has been captured correctly as shown in Fig. 6(b). A more quantitative comparison has been made by plotting the vertical (horizontal) velocity component profiles alone the horizontal (vertical) lines across the primary vortex centers [37] in Fig. 7. The agreements are quite satisfactory considering that dugksFoam used much less physical space cells and discrete velocity points.

4.3. Lid-driven cubic cavity flow

In this test, we apply the dugksFoam to a three dimensional low-speed flow simulation in the transition regime and compare the results with the DSMC solution. The flow geometry is illustrated in Fig. 8. The size of the cubic cavity is $L = 1$m. The lid (top boundary) of the cavity moves in the positive X direction with a constant velocity $U_w = 50$m/s, while the other walls are kept fixed. All of the sides are assumed to be purely diffusive walls and are kept at a uniform temperature $T_w = 273$K. The gas in the cavity is argon with molecular mass $m = 6.63 \times 10^{-26}$kg and diameter $d = 4.17 \times 10^{-10}$m. The gas viscosity depends on the temperature by $\mu = \mu_{\text{ref}}(T/T_w)^{\omega}$ with $\omega = 0.81$, corresponding to the variable hard sphere (VHS) model of the argon molecules interaction [1]. The reference viscosity μ_{ref} is the calculated by [1]

$$\mu_{\text{ref}} = \frac{15}{2}\frac{(mk_BT_w/\pi)^{1/2}}{(5-2\omega)(7-2\omega)d^2}, \tag{24}$$

where k_B is the Boltzmann constant. The Knudsen number defined as $\text{Kn}_{\text{ref}} = \lambda_{\text{ref}}/L$ is 0.1, where the reference mean free path λ_{ref} is calculated from the initial uniform gas density ρ_0 by $\lambda_{\text{ref}} = m/(\sqrt{2}\pi d^2\rho_0)$ [1].

In the dugksFoam simulation, the physical space is divided non-uniformly into 36^3 hexahedrons. The cell size grades increasingly towards the cavity center with a cell-to-cell expansion ratio of 1.03. The three dimensional velocity space is discredited using 28 half-range Gauss-Hermit quadrature points in each direction. The CFL number is set to be 0.8. The numerical scheme for the gradient evaluation is the unlimited Gauss linear scheme. The DSMC solution is obtained from the open source dsmcFoam code [26]. The dsmcFoam is also developed in the OpenFOAM framework and has be verified thoroughly in the literature [26]. In the DSMC simulation, a uniform mesh with 40^3 cells is used. Initially, 50 DSMC particles is placed in each cell. VHS model is used for the gas molecular interaction. The time step size is fixed at 1.6426×10^{-5}s. The sampling of the steady state result begins at physical time 20s and ends at 74s.

Fig. 9 shows the temperature contours predicted by dugksFoam and dsmcFoam. It can be seen that the two results agree well in general, even though the DSMC solution exhibits strong fluctuation. To compare the two solutions more precisely, we present the temperature distributions on the OXY plane and OZY plane as well as the X and Y components of the velocity (U and

V) distributions on the OXY plane in Fig. 10, from which, we can observe that the velocity field predicted by dugksFoam matches accurately with that of dsmcFoam. Regarding the temperature distributions, the dugksFoam result agrees with that of dsmcFoam on the whole but obvious differences can be observed in the up corners. The differences can be explained as follows. Firstly, dugksFoam uses the Shakhov model equation, while the dsmcFoam uses the full Boltzmann collision kernel. The two different approximations of the molecular interaction can leads to difference in the temperature field for externally driven flows [38, 39]. Secondly, there are strong statistic noises in the DSMC solution despite long time averaging has been done before outputting the results.

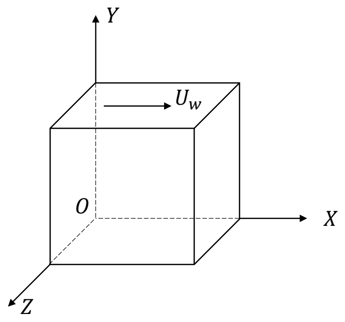
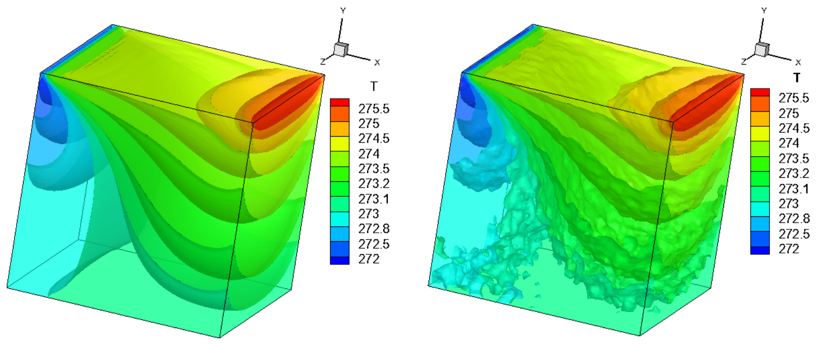


Figure 8: Illustration of the lid-driven cubic cavity flow.

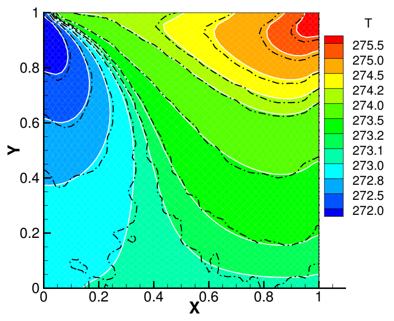


(a) (b)

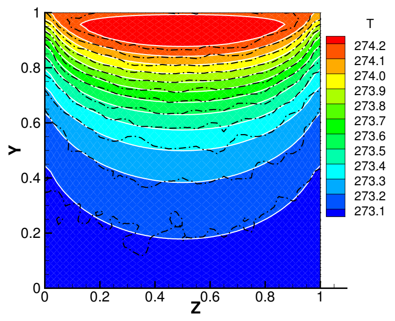
Figure 9: Temperature iso-surfaces of the cubic cavity case. (a) dugksFoam. (b) dsmcFoam.

5. Parallel efficiency

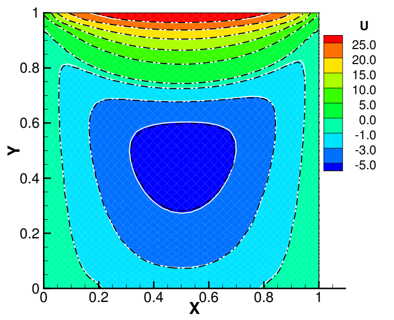
We now assess the parallel performance of dugksFoam. Several factors affecting the performance will be identified. The performance difference using



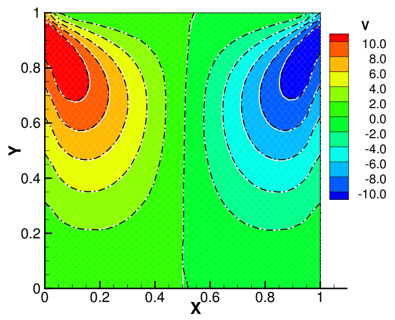
(a)



(b)



(c)



(d)

Figure 10: Temperature and velocity distribution in different cut-planes of the cubic cavity. Dashed black lines: dsmcFoam. While lines with colored background: dugksFoam. (a) Temperature in symmetric XY plane. (b) Temperature in symmetric ZY plane. (c) U-velocity in symmetric XY plane. (d) V-velocity in symmetric XY plane.

the two different domain decomposition approaches, i.e., the PSD and the VSD (Sec. 3.3) will be analyzed.

The test problems are the two-dimensional and three-dimensional lid-driven cavity flows. The testing platform is a small cluster with 16 computing nodes and 1 managing/IO node. Each computing node has two Intel E5-2680v3 Xeon (Haswell) CPUs and 64GB DDR4 memory operating at a frequency of 2133MHz. Each CPU has 12 cores and works at clock frequency 2.50GHz. The computing nodes are inter-connected by a fourteen data rate (FDR) InfiniBand network. Only 11 computing nodes and a maximum of 256 cores will be used in this test. Both OpenFOAM and dugksFoam are compiled using Intel C/C++ compiler of version 15.0.1 with `-O3` optimization flag, and are linked to the Intel MPI library with version 5.0. For each of the test problem, the solver runs using different numbers of cells and discrete velocities. The maximum problem size is limited by the total memory available. The average computing time for one step of evolution using different MPI processes are measured. The total running time (wall clock time) is insured to be over 100s. The IO time is not counted. The MPI processes are spawned among as much number of nodes as possible to alleviate the memory accessing pressure.

Parallel speedups against serial run using the PSD and VSD approaches are calculated and shown in Fig. 11 and Fig. 12, respectively. The ideal speedup curves corresponding to the linear speedup have been included. It is noted that for the case with 60^3 cells and 24^3 discrete velocities, the solver is unable to run with only 4 or less computing nodes due to the limited total memory. So the actual running time using 1 to 4 cores are extrapolated from that of 8 cores assuming linear speedup there. From Fig. 11, we can see the scaling performance is rather poor for two dimensional simulations using the PSD approach. For three dimensional simulations the scaling performance improves significantly but only if using larger number (60^3) of cells. The maximum speedup observed is 67.1 and achieves at the cases of 60^3 cell and 24^3 discrete velocities using 128 cores. Comparing Fig. 12 and Fig. 11, we can observe that using the VSD approach, the scaling performance increases considerably, especially for two dimensional cases and three dimensional cases with small number of cells. The maximum speedup observed is 109.2 achieving at the case of 30^3 cells and 45^3 discrete velocities using 256 cores.

Fig. 12 also shows that scaling performance deteriorates at smaller numbers of discrete velocities. The reason is that the pure computing time scales linearly with the number of discrete velocities but the communication time is unchanged when using the VSD approach. The performance here is obviously limited by the global reductions of the moments which are natural global barriers for the MPI processes. Another interesting phenomenon observed in Fig. 12 is that the scaling performance turns out to be insensitive to the number of cells, which means the bottleneck here is the second factor for the VSD approach analysed in Sec. 3.3.2, i.e., the global reduction efficiency is low if using large number of MPI processes.

Overall, the VSD is the preferred choice to run the solver in parallel, at least for typical numbers of cells and discrete velocities. However, it should be noted

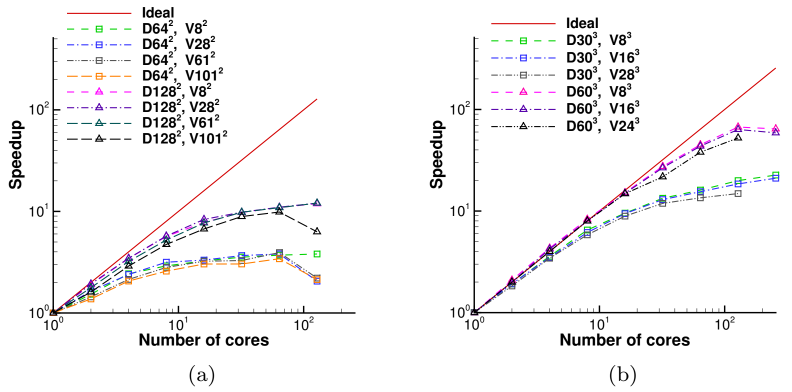


Figure 11: Parallel speedups using the physical space decomposition (PSD) approach. (a) Two dimensional cases. (b) Three dimensional cases. DM^d and VN^d mean using M^d cells and N^d discrete velocities, respectively.

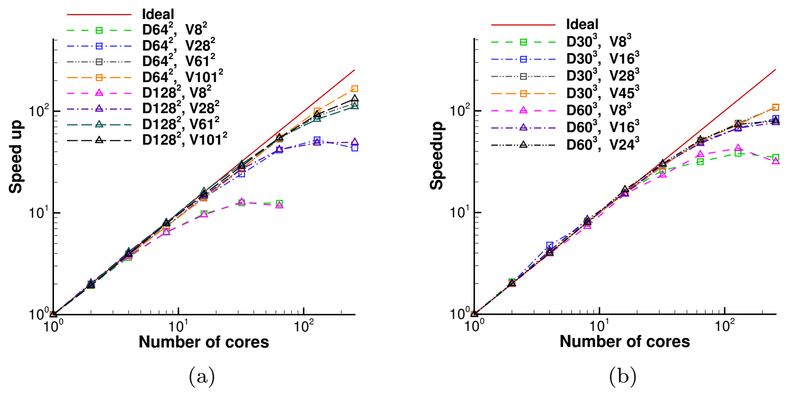


Figure 12: Parallel speedups using the velocity space decomposition (VSD) approach. (a) Two dimensional cases. (b) Three dimensional cases. DM^d and VN^d mean using M^d cells and N^d discrete velocities, respectively.

that, the PSD based MPI communication in the current solver is inefficient due to the calling of MPI send/receive operations for each discrete velocity, as have been explained in Sec. 3.3.1. A more sophisticated implementation of the PSD can improve the parallel performance, and even better than the VSD. Such as in Ref. [14], by using the asynchronous, non-blocking and bundled MPI send/receive MPI communications, the parallel efficiency using the PSD can be better than the VSD approach.

6. Conclusion and further improvements

An open source deterministic solver for the Boltzmann equation with Shakhov model has been developed in the OpenFOAM framework based on the recently proposed DUGKS method. The solver has been validated using several non-equilibrium flow cases including even three dimensional one. The results are in good agreement with either DSMC results or data in literatures.

The velocity space decomposition based MPI parallel computing features has been developed in addition to the physical space decomposition based one that shipped with the official OpenFOAM release. The parallel performance evaluations demonstrated that the newly implemented velocity space decomposition approach offers a much better scaling ability than the physical space decomposition based approach. Speedup by two orders of magnitude can be achieved using 256 cores on a small cluster for both two and three dimensional simulations.

The solver inherits many advantages of the OpenFOAM framework such as the arbitrary unstructured mesh ability [30] and abundant pre- and post-processing utilities. The deterministic nature and asymptotical preserving feature of the DUGKS method make this solver competitive in simulating low-speed flows in transition and near continuum regimes compared with other solvers based on the popular DSMC method.

Further optimizations or extensions can be made on this solver to make it more efficient or general. For instance, it is expected that a hybrid physical space and velocity space decomposition method can offer much better parallel efficiency than the pure velocity space decomposition approach. In addition, the current solver can be optimized for a better pure computing (serial) efficiency.

Acknowledgments

The present work is supported by the National Key Research Project (No. 2016YFB0600805), National Science Foundation of China (Grant No. 11602091), and Foundation for Student Innovation and Venture of the Huazhong University of Science and Technology (Grant No. 20150611).

Reference

[1] G. A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Clarendon Press, 1994.

[2] J. E. Broadwell, “Study of rarefied shear flow by the discrete velocity method,” *Journal of Fluid Mechanics*, vol. 19, no. 03, pp. 401–414, 1964.

[3] V. V. Aristov, *Direct methods for solving the Boltzmann equation and study of nonequilibrium flows.* Springer Science & Business Media, 2001, vol. 60.

[4] L. Mieussens, “Discrete-velocity models and numerical schemes for the Boltzmann-BGK equation in plane and axisymmetric geometries,” *Journal of Computational Physics*, vol. 162, no. 2, pp. 429–466, 2000.

[5] T. Inamuro and B. Sturtevant, “Numerical study of discrete-velocity gases,” *Physics of Fluids A: Fluid Dynamics (1989-1993)*, vol. 2, no. 12, pp. 2196–2203, 1990.

[6] T. Ohwada, Y. Sone, and K. Aoki, “Numerical analysis of the poiseuille and thermal transpiration flows between two parallel plates on the basis of the Boltzmann equation for hard-sphere molecules,” *Physics of Fluids A*, vol. 1, no. 12, pp. 2042–2049, 1989.

[7] T. Ohwada, “Structure of normal shock waves: Direct numerical analysis of the Boltzmann equation for hard-sphere molecules,” *Physics of Fluids A: Fluid Dynamics (1989-1993)*, vol. 5, no. 1, pp. 217–234, 1993.

[8] Y. Sone, *Molecular Gas Dynamics: Theory, Techniques, and Applications.* Birkhäuser Basel, 2007.

[9] J. Y. Yang and J. C. Huang, “Rarefied flow computations using nonlinear model Boltzmann equations,” *Journal of Computational Physics*, vol. 120, no. 2, pp. 323–339, 1995.

[10] L. Mieussens, “A survey of deterministic solvers for rarefied flows,” in *Proceedings of the 29th International symposium on Rarefied Gas Dynamics*, vol. 1628, 2014, p. 943.

[11] K. Xu, *Direct Modeling for Computational Fluid Dynamics*, ser. Advances in Computational Fluid Dynamics. World Scientific Publishing, 2015.

[12] F. Filbet and S. Jin, “A class of asymptotic-preserving schemes for kinetic equations and related problems with stiff sources,” *Journal of Computational Physics*, vol. 229, no. 20, pp. 7625–7648, 2010.

[13] Y. Zhu, C. Zhong, and K. Xu, “Implicit unified gas-kinetic scheme for steady state solutions in all flow regimes,” *Journal of Computational Physics*, vol. 315, no. 15, pp. 16–38, 2016.

[14] V. Titarev, M. Dumbser, and S. Utyuzhnikov, “Construction and comparison of parallel implicit kinetic solvers in three spatial dimensions,” *Journal of Computational Physics*, vol. 256, no. 1, pp. 17–33, 2014.

[15] C. K. Chu, “Kinetic-theoretic description of the formation of a shock wave,” *Physics of Fluids*, vol. 8, no. 1, pp. 12–22, 1965.

[16] P. L. Bhatnagar, E. P. Gross, and M. Krook, “A model for collision processes in gases. I. small amplitude processes in charged and neutral one-component systems,” *Physical Review*, vol. 94, no. 3, p. 511, 1954.

[17] L. H. Holway Jr, “New statistical models for kinetic theory: methods of construction,” *Physics of Fluids (1958-1988)*, vol. 9, no. 9, pp. 1658–1673, 1966.

[18] E. M. Shakhov, “Generalization of the Krook kinetic relaxation equation,” *Fluid Dynamics*, vol. 3, no. 5, pp. 95–96, 1968.

[19] Z.-H. Li and H.-X. Zhang, “Gas-kinetic numerical studies of three-dimensional complex flows on spacecraft re-entry,” *Journal of Computational Physics*, vol. 228, no. 4, pp. 1116–1138, 2009.

[20] V. I. Kolobov, R. R. Arslanbekov, V. V. Aristov, A. A. Frolova, and S. A. Zabelok, “Unified solver for rarefied and continuum flows with adaptive mesh and algorithm refinement,” *Journal of Computational Physics*, vol. 223, no. 2, pp. 589–608, 2007.

[21] V. A. Titarev, “Efficient deterministic modelling of three-dimensional rarefied gas flows,” *Communications in Computational Physics*, vol. 12, no. 1, p. 162, 2012.

[22] A. Frezzotti, G. P. Ghiroldi, and L. Gibelli, “Solving model kinetic equations on gpus,” *Computers & Fluids*, vol. 50, no. 1, pp. 136–146, 2011.

[23] Y. Y. Kloss, P. V. Shuvalov, and F. G. Tcheremissine, “Solving Boltzmann equation on GPU,” *Procedia Computer Science*, vol. 1, no. 1, pp. 1083–1091, 2010.

[24] S. Chigullapalli and A. Alexeenko, “Unsteady 3d rarefied flow solver based on Boltzmann-ESBGK model kinetic equations,” in *41st AIAA Fluid Dynamics Conference and Exhibit. AIAA*, 2011.

[25] C. Baranger, J. Claudel, N. Hérouard, and L. Mieussens, “Locally refined discrete velocity grids for stationary rarefied flow simulations,” *Journal of Computational Physics*, vol. 257, Part A, pp. 572–593, 2014.

[26] T. J. Scanlon, E. Roohi, C. White, M. Darbandi, and J. M. Reese, “An open source, parallel dsmc code for rarefied gas flows in arbitrary geometries,” *Computers & Fluids*, vol. 39, no. 10, pp. 2078–2089, 2010.

[27] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, “A tensorial approach to computational continuum mechanics using object-oriented techniques,” *Computers in Physics*, vol. 12, no. 6, pp. 620–631, 1998.

[28] OpenFOAM, *OpenFOAM, The Open Source CFD Toolbox, User Guide*, 2nd ed. OpenCFD Ltd., 2015.

[29] Z. Guo, R. Wang, and K. Xu, “Discrete unified gas kinetic scheme for all Knudsen number flows. II. thermal compressible case,” *Physical Review E*, vol. 91, no. 3, p. 033313, 2015.

[30] L. Zhu, Z. Guo, and K. Xu, “Discrete unified gas kinetic scheme on unstructured meshes,” *Computers & Fluids*, vol. 127, pp. 211–225, 2016.

[31] J. Blazek, *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[32] E. Sozer, C. Brehm, and C. C. Kiris, “Gradient calculation methods on arbitrary polyhedral unstructured meshes for cell-centered cfd solvers,” in *52nd Aerospace Sciences Meeting, no. AIAA*, vol. 1440, 2014, Conference Proceedings.

[33] M. Aftosmis, D. Gaitonde, and T. S. Tavares, “Behavior of linear reconstruction techniques on unstructured meshes,” *AIAA journal*, vol. 33, no. 11, pp. 2038–2049, 1995.

[34] D. J. Mavriplis, “Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes,” *AIAA paper*, vol. 3986, p. 2003, 2003.

[35] OpenFOAM, *OpenFOAM, The Open Source CFD Toolbox, Programmer’s Guide*, 2nd ed. OpenCFD Ltd., 2015.

[36] A. A. AlOnazi, “Design and optimization of openfoam-based CFD applications for modern hybrid and heterogeneous hpc platforms,” Thesis, King Abdullah University of Science and Technology, 2014.

[37] M. Vargas, G. Tatsios, D. Valougeorgis, and S. Stefanov, “Rarefied gas flow in a rectangular enclosure induced by non-isothermal walls,” *Physics of Fluids (1994-present)*, vol. 26, no. 5, p. 057101, 2014.

[38] J. C. Huang, K. Xu, and P. B. Yu, “A unified gas-kinetic scheme for continuum and rarefied flows II: Multi-dimensional cases,” *Communications in Computational Physics*, vol. 12, no. 3, pp. 662–690, 2012.

[39] C. Liu, K. Xu, Q. Sun, and Q. Cai, “A unified gas-kinetic scheme for continuum and rarefied flows IV: Full Boltzmann and model equations,” *Journal of Computational Physics*, vol. 314, pp. 305–340, 2016.