

A multi-level parallel solver for rarefied gas flows in porous media

Minh Tuan Ho^{a,1}, Lianhua Zhu^{a,b,1}, Lei Wu^a, Peng Wang^a, Zhaoli Guo^b, Zhi-Hui Li^{c,d}, Yonghao Zhang^{a,*}

^a James Weir Fluids Laboratory, Department of Mechanical and Aerospace Engineering, University of Strathclyde, Glasgow G1 1XJ, UK

^b State Key Laboratory of Coal Combustion, School of Energy and Power, Huazhong University of Science and Technology, Wuhan, 430074, China

^c Hypervelocity Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang 621000, China

^d National Laboratory for Computational Fluid Dynamics, No. 37 Xueyuan Road, Beijing 100191, China

ARTICLE INFO

Article history:

Received 16 November 2017

Received in revised form 18 May 2018

Accepted 20 August 2018

Available online 30 August 2018

Keywords:

Rarefied gas dynamics

Porous media

Multi-level parallel

Permeability

ABSTRACT

A high-performance gas kinetic solver using multi-level parallelization is developed to enable pore-scale simulations of rarefied flows in porous media. The Bhatnagar–Gross–Krook model equation is solved by the discrete velocity method with an iterative scheme. The multi-level MPI/OpenMP parallelization is implemented with the aim to efficiently utilize the computational resources to allow direct simulation of rarefied gas flows in porous media based on digital rock images for the first time. The multi-level parallel approach is analyzed in detail confirming its better performance than the commonly-used MPI processing alone for an iterative scheme. With high communication efficiency and appropriate load balancing among CPU processes, parallel efficiency of 94% is achieved for 1536 cores in the 2D simulations, and 81% for 12288 cores in the 3D simulations. While decomposition in the spatial space does not affect the simulation results, one additional benefit of this approach is that the number of subdomains can be kept minimal to avoid deterioration of the convergence rate of the iteration process. This multi-level parallel approach can be readily extended to solve other Boltzmann model equations.

© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Gas transport in ultra-tight porous media has recently received extensive attention due to its important role in extracting unconventional gas resources and developing new technologies such as fuel cells, filters and catalytic converters [1,2]. However, understanding of gas transport in ultra-tight porous materials remains a research challenge for theoretical, computational and experimental studies. As the pore space can be as small as a few nanometers, the conventional continuum flow theories such as the Darcy law and the Klinkenberg model become invalid. Meanwhile, the low permeability (at the nano-Darcy scale) is difficult for experimental measurements [3]. Therefore, direct pore-scale simulation of gas flow in ultra-tight porous material becomes even more important to unravel complex gas transport mechanisms, which can also provide reliable data for upscaling. For example, the apparent permeability obtained at the representative elementary volume scale can be plugged into macroscopic upscaling equations for predicting gas production [4].

In ultra-tight porous media, gas rarefaction effect becomes important to gas transport as the molecular mean free path is

comparable to the average pore size. The rarefaction level can be indicated by the Knudsen number Kn , defined as the ratio of the molecular mean free path λ to the characteristic length L of the flow, i.e.

$$Kn = \frac{\lambda}{L}, \quad \lambda = \frac{\mu(\hat{T}_0)}{\bar{p}} \sqrt{\frac{\pi R \hat{T}_0}{2}}, \quad (1)$$

where R , \bar{p} and $\mu(\hat{T}_0)$ are the specific gas constant, the mean gas pressure and dynamic viscosity of gas at a reference temperature \hat{T}_0 , respectively. Gas flows can be roughly categorized into four regimes: continuum ($Kn < 0.001$), slip ($0.001 < Kn < 0.1$), transition ($0.1 < Kn < 10$) and free-molecular ($Kn > 10$) flow regimes. The Navier–Stokes equations are applicable for continuum flow regime and their validity might be extended to the slip flow regime by introducing velocity-slip boundary condition at the solid surfaces [5,6]. Further increasing rarefaction level into the transition and free-molecular flow regimes, the linear constitutive relations as assumed in the Navier–Stokes equations are no longer valid [7]. Consequently, gas kinetic equations such as the Boltzmann equation or its model equations should be used instead of the Navier–Stokes equation.

The Boltzmann equation for rarefied gas flows, describes the temporal \hat{t} and spatial $\hat{\mathbf{x}}$ evolution of the distribution function

* Corresponding author.

E-mail address: yonghao.zhang@strath.ac.uk (Y. Zhang).

¹ Both authors contributed equally.

$\hat{f}(\hat{\mathbf{x}}, \hat{\mathbf{v}}, \hat{t})$ of gas molecules with the velocity $\hat{\mathbf{v}}$

$$\frac{\partial \hat{f}}{\partial \hat{t}} + \hat{\mathbf{v}} \cdot \frac{\partial \hat{f}}{\partial \hat{\mathbf{x}}} = I(\hat{f}, \hat{f}_*), \quad (2)$$

where the Boltzmann collision integral $I(\hat{f}, \hat{f}_*)$ represents the rate at which the distribution function varies before (\hat{f}) and after (\hat{f}_*) collisions. Numerical solutions of this nonlinear integro-differential equation for realistic problems are tremendously expensive, in terms of computational cost and memory requirement, which is due to the complex five-fold integral I as well as multidimensional phase space (3D spatial domain Ω_x , 3D velocity domain Ω_v , and 1D temporal domain Ω_t for unsteady flows). The full Boltzmann solvers can be classified into stochastic approach, such as the direct simulation Monte Carlo (DSMC) method, and deterministic approach, such as discrete velocity method (DVM) and fast spectral method (FSM) [8–14]. In order to obtain reasonable approximation of the Boltzmann equation, the full collision integral I is commonly simplified by a relaxation-time collision model, e.g. Bhatnagar–Gross–Krook (BGK) [15], ellipsoidal-statistical BGK (ES-BGK) [16] or Shakhov (S) [17] model. The Boltzmann model equations preserve the main physical properties, e.g. conservation of mass, momentum and energy, and their accuracy in modeling rarefied gas flows has been assessed. The models have been used for a wide range of applications from high-altitude aerodynamics to microfluidics. Among various numerical methods for the Boltzmann model equations, lattice Boltzmann model (LBM) [18] is well developed and widely used for modeling flows in porous media thanks to easy implementation of boundary condition on complex surfaces [19,20]. However, the conventional LBM fails to capture rarefaction effects in the early transition flow regime even for a simple Poiseuille flow due to a finite number of velocity grid points N_v [21,22]. It is demonstrated that high-order LBM is needed to capture the Knudsen paradox phenomena in a straight channel [23,24]. Therefore, the accuracy of conventional LBM for porous media flows in the transition and free-molecular regimes is still questionable and other gas kinetic methods are required to simulate rarefied gas flows in these regimes. As the DSMC method is impractical for often low-speed flows in porous media, the DVM is adopted here.

Even with the most simple collision model, i.e. the BGK equation, multi-dimensional phase space still prohibits practical simulations from serial computing. For example, a serial calculation of a porous sample of 1000^3 -voxel image may need several terabytes of memory and years of the wall-clock time, even with a modest molecular velocity grid of $N_v = 10^3$. In the last decade, rapid advances in massively parallel computing technology have paved the way for gas kinetic solvers for direct simulations of flows in porous media [25–33]. For calculations on a single computing node, kinetic solvers are usually parallelized using Open Multi-Processing (OpenMP) or Graphical Processing Unit (GPU). The OpenMP of shared memory model is accomplished by multi-threaded parallelism with the advantages of simplicity, portability and extensibility. However, scalability of OpenMP is limited by the number of cores on a single computing node. For example, speedup of 46 times is reported for 3D Sod problem executed on a decent 64-core machine [26]. For applications that seek floating point performance, GPU is much faster than the traditional CPU (Central Processing Unit). The GPU-based BGK solver may provide impressive acceleration (about 600 and 340 times for the 1D shock wave and 2D driven cavity tests, respectively) thanks to hundreds or thousands of “light weight” stream processors of GPU [25]. Nonetheless, it is normally restricted to relatively small domain mainly due to limited on-board global memory, e.g. a total grid points of up to $N_x \cdot N_v = 25.6 \times 10^3 \cdot 8 \times 10^3$ is reported in the above study.

For large scale calculations, the computing nodes are interconnected by high-speed networks and the Message Passing Interface (MPI) library is usually utilized for communication between subdomains, which are decomposed in either the velocity domain Ω_v or the spatial domain Ω_x . The Ω_v -decomposition is simple to implement and easy to achieve load balancing, thus it is usually favored. For instance, in calculation of 3D hypersonic flows around space vehicles, in which high resolution and a broad velocity space are desired, the Ω_v -decomposition strategy enables high efficiency of 88% for 20 000-core computing (in comparison with the 500-core one) with the total grid points of $191 \times 10^3 \cdot 729 \times 10^3$ [31]. Nevertheless, large amount of communications in calculating macroscopic parameters (i.e. the moments of distribution function \hat{f} over the velocity domain Ω_v) at every N_x spatial grid points reduce the performance significantly, especially for a large N_x . Hence Ω_x -decomposition is preferred (or inevitable) when the spatial domain is large compared to the velocity domain. For example, the simulations of 3D pressure-driven flow in a pipe, which uses nearly $350 \times 10^3 \cdot 4.1 \times 10^3$ grid points, have parallel efficiencies of 96% for Ω_x -decomposition and 70% for Ω_v -decomposition using 512 cores (in comparison with the 64 cores) [28]. Regardless of Ω_x -decomposition or Ω_v -decomposition, pure MPI parallelization has limited scalability for a fixed number of grid points $N_x \cdot N_v$. The subdomain handled by each MPI core becomes smaller whereas the total amount of MPI communication increases with the number of core N_c , leading to load unbalancing. One can observe significant drop of efficiency when the number of cores doubles to 1024 in the above example, i.e. from 96% to 64% and from 70% to 15% respectively. Performance comparison between the two strategies of domain decomposition can be found in Ref. [33].

The limitation on MPI scalability can be mitigated by introducing the second level parallelization. More precisely, two-level MPI/OpenMP parallel approach, in which the top-level MPI process is responsible for each Ω_x -subdomain, and a group of OpenMP threads at the bottom-level accelerate computing within the Ω_x -subdomain as well as communicating with other Ω_x -subdomains. This two level approach will help to achieve load balancing and reduce memory consumption for duplicated data [34]. To the best of our knowledge, this multi-level parallel approach for a gas kinetic solver was first proposed by [32]. However, the tested largest domain is relatively small with the total grid points of $50 \times 10^3 \cdot 3 \times 10^3$ and the parallel performance was not analyzed in that study. A typical modern micro-CT scanner can provide images with 2000^3 voxels and advanced synchrotron imaging may allow much larger images to be reconstructed [19]. As a result, the two-level parallel MPI/OpenMP approach with excellent scalability is required for the kinetic solver to be able to directly simulate gas flows in the porous media with flow passages provided by high-resolution images.

Apart from parallel scalability, convergence rate to steady state also contributes to the total simulation time. To solve the BGK equation, the most numerical schemes are designed for unsteady flows [25,26,30,33,35–39]. A few BGK solvers recently employ implicit time-marching scheme to accelerate steady state solutions with a large Courant–Friedrichs–Lewy (CFL) number [27,32,40]. It is demonstrated in Refs. [41] and [28] that domain decomposition in the spatial space Ω_x deteriorates, to some extent, the convergence rate of implicit time-marching schemes. Taking into account a smaller number of subdomains for the same number of cores, multi-level parallel approach may mitigate deterioration in the convergence rate for pure MPI Ω_x -decomposition.

The present work is therefore to develop a multi-level parallel BGK kinetic solver with enhanced scalability for steady rarefied gas flows in porous media. The solver is designed to efficiently solve flows in porous media with complex structures described by binary 2D slide-images (composed of pixels) or 3D volume-images (composed of voxels). Parallel scalability of two-level MPI/OpenMP approach is analyzed and compared with that of the pure MPI approach.

2. Governing equation and numerical method

2.1. Governing equation and its boundary conditions

The BGK model equation can be employed to describe low-speed gas flows in ultra-tight porous media, where the flow resistance is high. The distribution function is linearized in the standard manner as $\hat{f} = \hat{f}_{eq}(1 + h)$ [42], and the Maxwellian distribution function is

$$\hat{f}_{eq} = \frac{\hat{n}_{eq}}{(2\pi R\hat{T}_0)^{3/2}} \exp\left(-\frac{|\hat{\mathbf{v}}|^2}{2R\hat{T}_0}\right). \quad (3)$$

The global equilibrium number density \hat{n}_{eq} is related to the mean gas pressure by the ideal gas law $\hat{n}_{eq} = \bar{p}/mR\hat{T}_0$, in which m is the molecular mass. The (dimensionless) perturbed distribution function $h(\mathbf{x}, \mathbf{v})$ is governed by the linearized BGK equation

$$\mathbf{v} \cdot \frac{\partial h}{\partial \mathbf{x}} = \frac{\sqrt{\pi}}{2Kn} \left[\varrho + 2\mathbf{u} \cdot \mathbf{v} + \tau \left(|\mathbf{v}|^2 - \frac{3}{2} \right) - h \right], \quad (4)$$

where time-dependent derivative in Eq. (2) is omitted as only steady-state solution is of interest. The following dimensionless quantities (denoted by omitting “hat” in the notations of the corresponding dimensional ones) are used

$$\mathbf{x} = \frac{\hat{\mathbf{x}}}{L}, \quad \mathbf{v} = \frac{\hat{\mathbf{v}}}{v_m}, \quad f = \frac{\hat{f}}{\hat{n}_{eq}/v_m^3}, \quad (5)$$

where $v_m = \sqrt{2R\hat{T}_0}$ is the most probable molecular speed. The perturbed number density ϱ , velocity \mathbf{u} and temperature τ are calculated as moments of perturbed distribution function h over the velocity space

$$\begin{aligned} \varrho &= \int f_{eq} h d\mathbf{v}, & \mathbf{u} &= \int \mathbf{v} f_{eq} h d\mathbf{v}, \\ \tau &= \frac{2}{3} \int |\mathbf{v}|^2 f_{eq} h d\mathbf{v} - \varrho, \end{aligned} \quad (6)$$

where the dimensionless equilibrium distribution function is

$$f_{eq} = \pi^{-3/2} \exp(-|\mathbf{v}|^2). \quad (7)$$

Boundary conditions for the BGK equation (4) should be specified at the solid surfaces and the outer faces of a porous medium. Gas-surface interaction is modeled by the Maxwell diffuse-specular reflection

$$h(\mathbf{v} | \mathbf{v} \cdot \mathbf{n} > 0) = \alpha \varrho_s(\mathbf{n}) + (1 - \alpha) h(\mathbf{v} - 2\mathbf{n}(\mathbf{v} \cdot \mathbf{n})), \quad (8)$$

where \mathbf{n} , ϱ_s , α are the outer normal unit vector of the solid surface, the perturbed gas number density on the solid surface and the tangential momentum accommodation coefficient (TMAC), respectively. Value of TMAC represents the diffuse portion of the reflected molecules, i.e. fully diffuse or fully specular reflections correspond to $\alpha = 1$ or $\alpha = 0$. This study uses the diffuse boundary condition $\alpha = 1$ at the solid surfaces. The perturbed gas number density on the solid surface is computed from the non-penetration condition, i.e. zero-mass flux through the solid surface

$$\varrho_s(\mathbf{n}) = -\frac{\int_{\mathbf{v} \cdot \mathbf{n} < 0} \mathbf{v} \cdot \mathbf{n} \exp(-|\mathbf{v}|^2) h d\mathbf{v}}{\int_{\mathbf{v} \cdot \mathbf{n} > 0} \mathbf{v} \cdot \mathbf{n} \exp(-|\mathbf{v}|^2) d\mathbf{v}}. \quad (9)$$

A porous medium can be constructed by the periodic replica of the representative elementary volume in the x_1 , x_2 , and x_3 directions, respectively. It is convenient to take the size of computational domain i.e. the representative elementary volume in the \hat{x}_1 direction as characteristic flow length $L = \hat{x}_1^{max} - \hat{x}_1^{min}$ in the definition of Knudsen number Eq. (1). At the inlet ($x_1 = x_1^{min}$) and

outlet ($x_1 = x_1^{max}$), periodic condition [43] representing the pressure gradient is applied for molecules entering the computational domain assuming that the pressure gradient only exists in the x_1 -direction, i.e.

$$\begin{aligned} h(x_1^{min}, x_2, x_3, v_1, v_2, v_3) &= (x_1^{max} - x_1^{min}) + h(x_1^{max}, x_2, x_3, v_1, v_2, v_3), \\ &\text{when } v_1 > 0, \\ h(x_1^{max}, x_2, x_3, v_1, v_2, v_3) &= (x_1^{min} - x_1^{max}) + h(x_1^{min}, x_2, x_3, v_1, v_2, v_3), \\ &\text{when } v_1 < 0. \end{aligned} \quad (10)$$

At the lateral faces of the porous medium, symmetric boundary conditions are implemented by the specular reflection, i.e.

$$\begin{aligned} h(x_1, x_2^{min}, x_3, v_1, v_2, v_3) &= h(x_1, x_2^{min}, x_3, v_1, -v_2, v_3), \\ &\text{when } v_2 > 0, \\ h(x_1, x_2^{max}, x_3, v_1, v_2, v_3) &= h(x_1, x_2^{max}, x_3, v_1, -v_2, v_3), \\ &\text{when } v_2 < 0, \\ h(x_1, x_2, x_3^{min}, v_1, v_2, v_3) &= h(x_1, x_2, x_3^{min}, v_1, v_2, -v_3), \\ &\text{when } v_3 > 0, \\ h(x_1, x_2, x_3^{max}, v_1, v_2, v_3) &= h(x_1, x_2, x_3^{max}, v_1, v_2, -v_3), \\ &\text{when } v_3 < 0. \end{aligned} \quad (11)$$

The dimensionless apparent permeability k , which is normalized by L^2 , is calculated as

$$k = \frac{Kn}{\sqrt{\pi}} \frac{(x_1^{max} - x_1^{min})^2}{(x_2^{max} - x_2^{min})(x_3^{max} - x_3^{min})} G_p, \quad (12)$$

where G_p is the dimensionless mass flow rate normalized by \bar{p}/v_m

$$G_p = 2 \int_{x_3^{min}}^{x_3^{max}} \int_{x_2^{min}}^{x_2^{max}} u_1(x_2, x_3) dx_2 dx_3. \quad (13)$$

In order to appropriately compare the pore-scale rarefaction effects in different porous samples of various sizes, the effective Knudsen number Kn^* is defined as

$$Kn^* = \frac{\lambda}{L^*} = \frac{Kn}{L^*/L}, \quad (14)$$

where the average pore size L^* is defined as

$$L^*/L = \sqrt{\frac{12k_\infty}{\epsilon}} \text{ for 2D, } L^*/L = \sqrt{\frac{8k_\infty}{\epsilon}} \text{ for 3D.} \quad (15)$$

In this study, permeability obtained with the smallest examined Kn , which ensures that flow is in the continuum regime, is considered as intrinsic permeability k_∞ to determine L^* . The porosity ϵ of a porous model in Eq. (15) is determined by the ratio of the number of fluid points to the total number of points, i.e. the percentage of voids in the digital images.

2.2. Discrete velocity method

DVM is one of the most common deterministic approaches to solve the Boltzmann equation and its simplified models [35,44], which projects the continuous molecular velocity space \mathbf{v} into a set of fixed N_v -discrete velocities $\mathbf{v}^{(k)}$ ($k = 1, 2, \dots, N_v$). Consequently, the BGK equation (4) is replaced by a system of N_v -independent equations. Generally speaking, choice of velocity grid, i.e. the number N_v and value of discrete velocity points $\mathbf{v}^{(k)}$, is problem dependent. The range of $\mathbf{v}^{(k)}$ must cover the scope of bulk velocity \mathbf{u} and temperature T in the whole flow domain, e.g. a broad

variety of \mathbf{u} or T requires a wide range of $\mathbf{v}^{(k)}$. The number N_v associates with accuracy order of numerical quadrature adopted for evaluating macroscopic parameters using Eq. (6). Highly rarefied (large Kn) flows usually demand a fine velocity grid to capture the discontinuity in the distribution function. The values of discrete velocity points are determined by the abscissas of the adopted quadrature rule. Detailed discussion on velocity grid can be found in Ref. [32]. In this study, the Cartesian velocity grid generated by half-range Gauss–Hermit quadrature [45] is employed.

2.3. Finite difference approximation of advection terms

As reconstructed digital images of porous media samples comprise of voxels, it is straightforward to construct uniform grids, where the fluid and solid points coincident with voids and matrix voxels, respectively. As a result, finite difference method (FDM) is convenient to approximate the advection terms on these uniform grids. The spatial derivatives in Eq. (4) are approximated by the upwind schemes, e.g. the gradient component of h at the fluid point \mathbf{x} projected in the x_j coordinate axis ($j = 1, 2, 3$) is evaluated as follows

$$\begin{aligned} \frac{\partial h(\mathbf{x}, \mathbf{v})}{\partial x_j} &\approx \mathcal{D}(h(\mathbf{x}, \mathbf{v}))_j = \text{sgn}(v_j)[C_0 h(\mathbf{x}, \mathbf{v}) \\ &+ C_1 h(\mathbf{x} - \text{sgn}(v_j)\Delta x \mathbf{i}_j, \mathbf{v}) \\ &+ C_2 h(\mathbf{x} - \text{sgn}(v_j)2\Delta x \mathbf{i}_j, \mathbf{v})], \end{aligned} \quad (16)$$

where sgn , Δx and \mathbf{i}_j are the sign (signum) function, the constant grid size and the unit vector of the x_j coordinate axis, respectively. The constants (C_0, C_1, C_2) are equal to $(1.5, -2, 0.5)/\Delta x$ for the second-order-accurate scheme, while they are $(1, -1, 0)/\Delta x$ for the first-order-accurate scheme. The second-order-accurate scheme is used by default while the first-order-accurate scheme is automatically deployed if the upstream grid point in the v_j -direction is located on the boundary surfaces. The upwind numerical stencils vary from 3 to 5 points for 2D flows and from 4 to 7 points for 3D problems. For instance, the full 5-point stencil and reduced 4-point stencil for a 2D flow problem are illustrated in Fig. 1 by the black dash frame around the central fluid point \mathbf{x} . To avoid checking the conditional expression IF–THEN(–ELSE) at every iteration, the order of accuracy associated with each spatial grid point is set in pre-processing by storing the values of (C_0, C_1, C_2) in an array. Therefore, we obtain a set of discretized equations by applied DVM and FDM on the BGK equation (4)

$$\begin{aligned} v_j^{(k)} \mathcal{D}(h(\mathbf{x}, \mathbf{v}^{(k)}))_j \\ = \frac{\sqrt{\pi}}{2Kn} \left[\varrho + 2u_j v_j^{(k)} + \tau \left(|\mathbf{v}^{(k)}|^2 - \frac{3}{2} \right) - h \right]. \end{aligned} \quad (17)$$

3. Kinetic solver for rarefied gas flow in porous media

3.1. Algorithmic procedure for the serial solver

The set of N_v -independent equations, i.e. Eq. (17) are solved by an iterative scheme, which is only suitable for structured spatial grids. Three major steps are to be completed successively in each iteration as follows:

- (i) Distribution function h in the *bulk region* is updated by solving Eq. (17).

To be more specific, each equation corresponding to a discrete velocity $\mathbf{v}^{(k)}$ can be independently solved by forward substitution along a path of sweep, namely raster scan, on all the fluid points in the spatial domain. The path of sweep is subject to numerical stencil and thus depends on the signs of

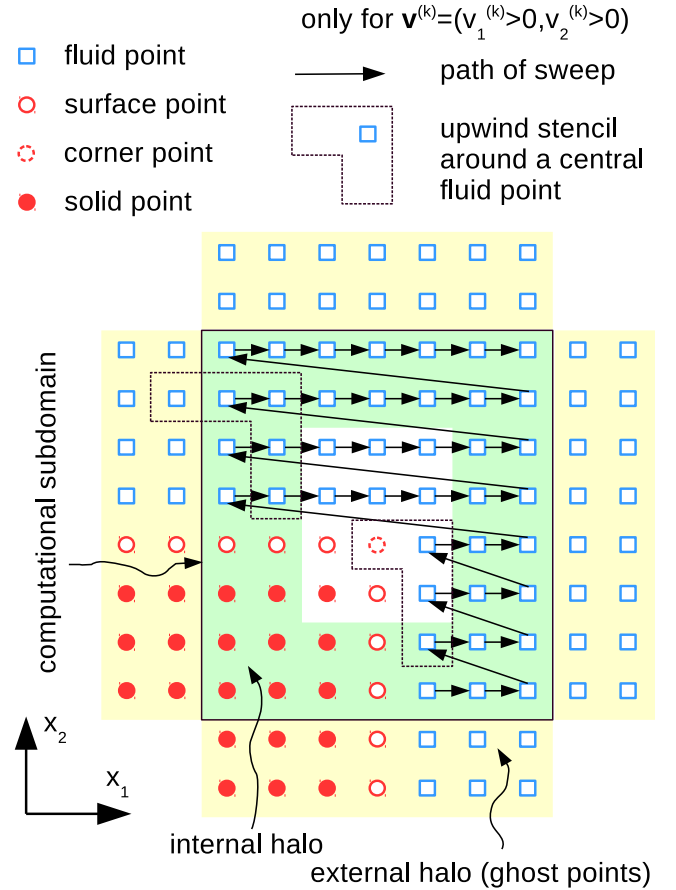


Fig. 1. The subdomain on the top-right quarter of the MPI 2×2 parallelization of the full domain of 14×16 pixels, in which a square cylinder of 8^2 pixels is located at the center. Internal halo/external halo (ghost points), in which data are copied to/duplicated from the neighboring subdomains (by MPI communication), are in the highlighted regions inside/outside the computational subdomain (bounded by the black solid rectangle). Un-highlighted region in the middle holds the data being local to each subdomain, i.e. no MPI communication with others. Forward substitution is performed sequentially by this stencil along a path of sweep, namely raster scan, on all the fluid points inside the computational domain. The upwind stencil (the black dash frame around the central fluid point) and path of sweep of discrete velocity $\mathbf{v}^{(k)} = (v_1^{(k)}, v_2^{(k)})$ are subjected to the signs of discrete velocity component, only illustrated for the velocity group of $v_1^{(k)} > 0, v_2^{(k)} > 0$.

all the components of the discrete velocity $v_j^{(k)}$. Fig. 1 demonstrates the path of sweep for the group of discrete velocity $v_j^{(k)} = (v_1^{(k)} > 0, v_2^{(k)} > 0)$ in a 2D simulation. This path starts on the bottom-left fluid point and then sweeps in the positive direction of v_1 . When it reaches the most right fluid point, it continues at the left most fluid point of the upward row in the positive direction of v_2 . Repeat the rightward and upward sweep until it reaches the top-right fluid point. One can imagine that a 3D porous structure can be created from this 2D porous structure by duplicating 2D slice in the positive direction of the x_3 coordinate axis. Considering the group of discrete velocity $v_j^{(k)} = (v_1^{(k)} > 0, v_2^{(k)} > 0, v_3^{(k)} > 0)$ in that 3D porous sample, the volume path of sweep can be developed from the above planar path of sweep by continuing from the bottom-left fluid point of the next plane in the positive direction of v_3 , and so forth. Paths of sweep for the other three (seven) groups of discrete velocity in 2D (3D) simulations can be deduced in an analogous manner. Distribution function of a fluid point is computed from the neighbor points in the upwind stencil, which have already

Algorithm 1: Two-level parallel MPI/OpenMP pseudo-code for one iteration. **MPI routines** and **OpenMP compiler directives** are in red and blue colors, respectively. By removing the highlighted lines, which represent work on data transfers between spatial subdomains, one obtains pure OpenMP pseudo-code. By discarding the blue lines, one obtains pure MPI pseudo-code with spatial domain decomposition. The serial pseudo-code corresponds to un-highlighted black lines.

```

data :  $h^{BC}$  and  $\varrho$ ,  $\mathbf{u}$ ,  $\tau$  obtained from the previous iteration or the initial conditions
results: updated  $h^{BC}$  and  $\varrho$ ,  $\mathbf{u}$ ,  $\tau$ 

1 !SOMP PARALLEL
2   !SOMP SINGLE
3     MPI_ISEND(sending_buffers); // nonblocking send
4     MPI_IRECV(receiving_buffers); // nonblocking receive
5   !SOMP DO // OpenMP threads share velocity domain  $\Omega_v$ 
6     Do  $k = 1$  to  $N_v$ 
7       Do  $i = 1$  to  $N_x$  // Raster scan on all fluid points (forward substitution), see Fig. 1
8       |  $h_{k,i} \leftarrow \text{FindH}(k, i)$ ; // Advection & collision Eqs.(16), (17)
9   !SOMP SINGLE
10    MPI_WAITALL; // wait for send & receive done
11  !SOMP DO // OpenMP threads share internal & external halo
12    Do  $i = 1$  to  $\text{halo}_{\max}$ 
13    |  $h^{\text{ext\_halo}} \leftarrow \text{receiving\_buffers}$ ; // buffer unpack
14    |  $\text{sending\_buffers} \leftarrow h^{\text{int\_halo}}$ ; // buffer pack
15  !SOMP DO // OpenMP threads share boundary points
16    Do  $i = 1$  to  $\text{Boundary}_{\max}$ 
17    |  $h_i^{BC} \leftarrow \text{FindBC}(i)$ ; // Boundary conditions Eqs.(8), (9),(10),(11)
18  !SOMP DO // OpenMP threads share fluid points
19    Do  $i = 1$  to  $N_x$ 
20    |  $\varrho_i, \mathbf{u}_i, \tau_i \leftarrow \text{FindMacro}(h_i)$ ; // Macroscopic parameters Eqs.(6)

```

been calculated either in some early stages of the current sweep (if the neighbors are the fluid points) or in the step (ii) of the previous iteration (if the neighbors are the boundary points). It should be noted that each solid corner point has either 2 or 3 outer normal unit vectors \mathbf{n}_j directing to its neighboring fluid points. If the numerical stencil includes a solid corner point that links to the central fluid point in the x_j -direction, diffuse-specular reflection at that corner point must be calculated with the normal vector \mathbf{n}_j by Eq. (8). For example, considering the numerical stencil of the fluid point on the right of the corner point in Fig. 1, diffuse-specular reflection at that corner point must be calculated with the normal vector $\mathbf{n}_2 = (1, 0)$.

- (ii) Applying the *boundary conditions*, i.e. Eq. (8) on solid surfaces, and updating the boundary conditions, i.e. Eqs. (10) and (11) on the outer faces of the porous sample.
- (iii) The *macroscopic flow fields* obtained from Eq. (6) and the gas number density on the solid surface from Eq. (9) are updated by the quadrature rule associated with the adopted velocity grid.

Only the resulting data for step (ii) and step (iii) are needed for the next iteration, thus we do not need to store the solved distribution function h in the bulk region as required in the time-marching schemes. The two supplementary steps are performed at every 100 iterations:

- (iv) *Permeability* k given by Eq. (12) is approximated by the trapezoidal rule applied on a transverse cross section.
- (v) The values of permeability obtained at the current iteration $k^{(l)}$ and the previous 100 iterations $k^{(l-100)}$ are used to assess *convergence*, i.e.

$$\text{Err}(l) = \frac{1}{100} \left| \frac{k^{(l)} - k^{(l-100)}}{k^{(l)}} \right| < 10^{-8}. \quad (18)$$

3.2. Two-level parallel implementation using MPI and OpenMP

The two-level parallel approach as illustrated in the Algorithm 1 is a natural option when the discretization in both the spatial and velocity spaces is required. At the upper level, the 3D/2D spatial domain Ω_x is decomposed into multiple cuboid/rectangular subdomains, and each subdomain is assigned to one MPI process. At the lower level, i.e. within each subdomain, OpenMP is employed to parallelize the major steps described in Section 3.1. Moreover, additional steps are introduced to deal with data transfer between subdomains, e.g. buffer packing/unpacking are also parallelized with OpenMP. In the rest of this paper, multi-level MPI/OpenMP parallelization will be denoted by MPI $c_1 \times c_2 \times c_3$ OMP c_4 . Here c_1, c_2, c_3 are the numbers of subdomain divisions along the x_1, x_2, x_3 coordinate axes, respectively, while c_4 is the number of OpenMP threads allocated to each subdomain. Therefore, the total number of subdomains, i.e. the MPI processes, is $c_1 \times c_2 \times c_3$ and the total number of adopted cores is $c_1 \times c_2 \times c_3 \times c_4$. Pure MPI or pure OpenMP parallelization corresponds to $c_4 = 1$ or $c_1 = c_2 = c_3 = 1$, respectively.

MPI communication occurs only between the spatial subdomains, see Fig. 1. To facilitate the overlapping of computation and MPI communication, we use the classical ghost points approach and the non-blocking version of the MPI send/receive subroutines. To be more specific, two layers of additional grid points, called ghost points or external halo, are padded around each computational subdomain, resulting in an extended subdomain. As a result, the fluid points near the communication boundaries, called internal halo, can be treated normally like the inner fluid points, although the numerical stencils around the internal halo may occupy points beyond the communication boundaries. Additional sending (receiving) buffers are allocated to store the distribution function of the out-going (in-coming) discrete velocities at each subdomain boundary. The non-blocking MPI send/receive-subroutines and the MPI wait-subroutine are called prior and posterior, respectively,

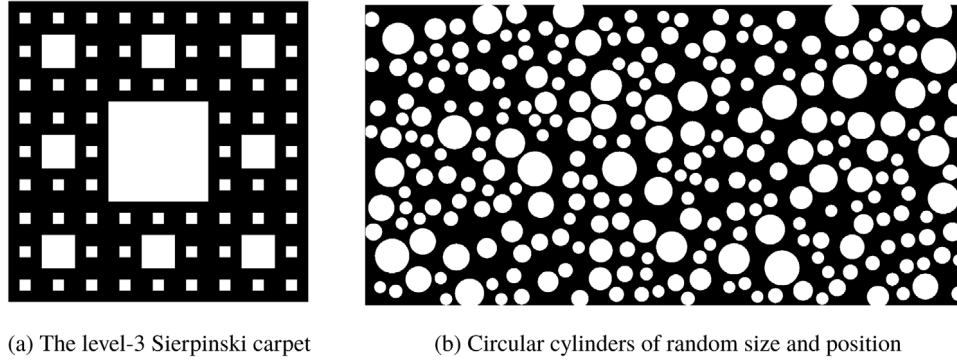


Fig. 2. 2D digital models of porous media, in which the white and black regions represent the matrix and voids, respectively. The spatial grid size N_x and porosity ϵ are (a) $N_x = 540 \times 540$, $\epsilon = 0.70$; and (b) $N_x = 3000 \times 1500$, $\epsilon = 0.60$.

Table 1

2D porous model of the Sierpinski carpet [Fig. 2(a)]: the wall clock time (in second) and the number of iterations (in curve parentheses) at various Kn with different spatial domain Ω_x -decomposition. The velocity grid size $N_v = 24^2$ and $N_v = 8^2$ are used for $Kn = 1$ and $Kn = 0.1, 0.01, 0.001$, respectively.

Kn ($Kn^* = 18Kn$)	0.001	0.01	0.1	1
MPI 1 \times 1 OMP 12	53.39 s (1900)	16.86 s (600)	8.43 s (300)	94.21 s (300)
MPI 2 \times 2 OMP 12	13.50 s (2200)	3.69 s (600)	1.84 s (300)	18.60 s (300)
MPI 4 \times 4 OMP 12	4.23 s (2500)	1.02 s (600)	0.87 s (500)	11.20 s (500)
k	1.781×10^{-4}	3.455×10^{-4}	2.246×10^{-3}	2.717×10^{-2}

to the processing of the bulk subdomains. Then, the out-going (incoming) distribution functions at the internal halo (external halo) are packed (unpacked) to (from) the sending (receiving) buffers. By using the two-level MPI/OpenMP parallel approach, the total number of MPI communication function calls is effectively reduced compared with the pure MPI approach, so that the communication congestion is avoided and the communication time is reduced so as to achieve MPI level optimization, in which the performance of the program is improved from the MPI level optimization and the algorithmic level optimization.

As stated in Section 1, the resulting parallel solver is not equivalent to the original serial solver in terms of the convergence history, because the grid points near the communication boundaries use the last iteration value of the upwind grid points. In the following Section, we will investigate how and to what extent the convergence rate is affected by the spatial domain decomposition, and even more importantly, whether the final permeability converges to the prediction of serial solver.

4. Convergence properties of the kinetic solvers

Both the 2D and 3D MPI/OpenMP parallel implementations of the current algorithm are compared against the OpenMP parallelization (without domain decomposition) in terms of the convergence rate and converged permeability. It is noted that the OpenMP parallelization has been validated for 2D flow through square array of circular cylinders in our previous work [46]. The term associated with the perturbed temperature τ in Eq. (17) has no influence on permeability in our test on isothermal flow thus it is neglected. The solvers are written in Fortran and compiled by the Cray Programming Environment (version 5.2.82), and run on the ARCHER super-computing system (118080 processing cores in total), the UK national academic HPC facility. Each compute node of the ARCHER contains two 12-core E5-2697 v2 (Ivy Bridge) series processors and 64 GB memory.

4.1. 2D flows

We first consider a regular model porous medium, i.e. the Sierpinski carpet, which is a famous plane fractal and can be constructed through recursion. The construction begins with a square.

The square is cut into 9 congruent sub-squares with a 3-by-3 grid, and the central sub-square is then removed. The same procedure is applied recursively to the remaining 8 sub-squares. We consider a three-level construction with a resolution of 540×540 pixels, see Fig. 2(a). The Knudsen number Kn is associated with the reference length $L = 540$ pixels taken from size of the sample in the x_1 direction, while the effective Knudsen number $Kn^* = 18.10Kn$ is estimated by Eq. (15) with $\epsilon = 0.70$ and $k_\infty = 1.781 \times 10^{-4}$. We consider Kn of 0.001, 0.01, 0.1 and 1, so the flow ranges from the slip to free-molecular flow regimes. The discrete velocity grids for the case of $Kn = 1$ are 24×24 , and for the others are 8×8 . For each Kn , we run the parallel solver with 2×2 and 4×4 even decomposition. The number of iterations and the wall clock time to reach the convergence criterion Eq. (18) for each kind of parallel decomposition are tabulated in Table 1. In the bottom row of the table, we also present the converged permeabilities obtained by the pure OpenMP computations. It is noted that pure OpenMP option (MPI 1 \times 1 OMP 12) predicts the identical results as the serial version (MPI 1 \times 1 OMP 1) in terms of convergence history and converged permeabilities where no domain decomposition is applied. The permeabilities predicted by the multi-level parallel are not shown here, because they match those of the pure OpenMP option, with the maximum relative deviation being 0.000203%. The convergence history of the permeabilities at different Knudsen numbers and domain decompositions are shown in Fig. 3(a). Both Table 1 and Fig. 3(a) show that the deterioration in convergence rate generally becomes worse with increasing spatial subdomains and Knudsen number. The number of iterations has risen by 32% and 67% for $Kn = 0.001$ and $Kn = 1$, respectively, when 16 subdomains are adopted. The slow convergence rate near the continuum regime for the classical DVM, without domain decomposition, has been confirmed by the Fourier stability analysis, where the spectral radius is found to be equal to unity [47]. Convergence rate dependency on Knudsen number was also analyzed for regular geometries, such as Poiseuille flow in plane channel and driven cavity flow, in Ref. [48].

Now we consider a more complex 2D porous media composed by circular cylinders with random locations and sizes, as shown in Fig. 2(b), where the porosity is 0.6. The spatial grid size is 3000×1500 , and $Kn = 0.001, 0.01, 0.1, 1$. The velocity grids

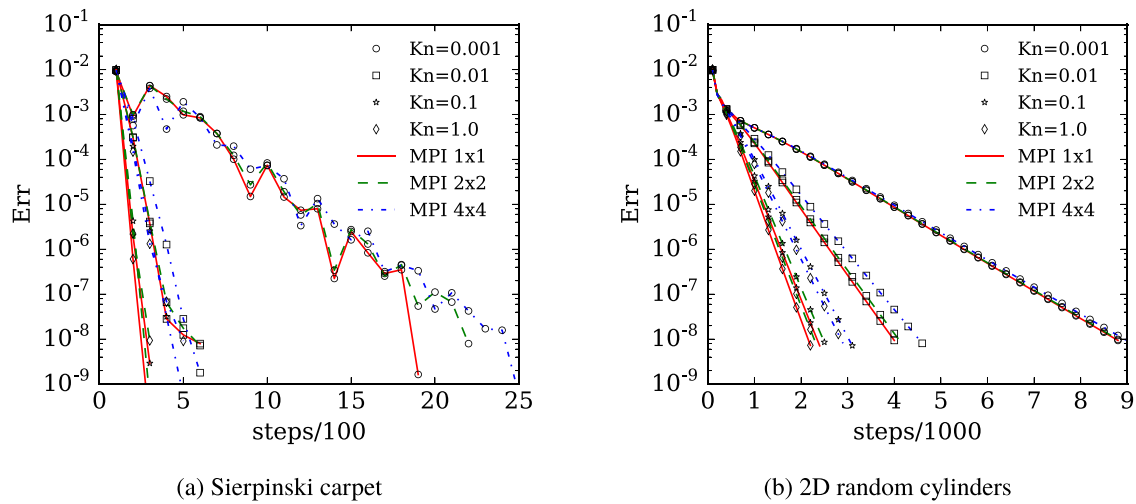


Fig. 3. Convergence history of the permeabilities given by Eq. (18) at different Knudsen numbers Kn with different spatial domain decomposition for the 2D models of porous media as shown in Fig. 2.

Table 2

The 2D porous model of random cylinders [Fig. 2(b)]: the wall clock time (in second) and the number of iterations (in curve parentheses) at various Knudsen number Kn with different spatial domain Ω_x -decomposition. The velocity grid size $N_v = 24^2$ and $N_v = 8^2$ are used for $Kn = 1, 0.1$ and $Kn = 0.01, 0.001$, respectively.

$Kn(Kn^* = 65Kn)$	0.001	0.01	0.1	1
MPI 1 × 1 OMP 12	2719 s (8800)	1236 s (4000)	8208 s (2400)	7524 s (2200)
MPI 2 × 2 OMP 12	730 s (8800)	344 s (4100)	2380 s (2500)	2160 s (2300)
MPI 4 × 4 OMP 12	234 s (9000)	120 s (4600)	791 s (3100)	740 s (2900)
k	1.201×10^{-5}	4.524×10^{-4}	3.758×10^{-3}	3.662×10^{-2}

Table 3

The 3D porous model of cubic sphere packing [Fig. 4(a)]: the wall clock time (in second) and the number of iterations (in curve parentheses) at various Knudsen number Kn with different spatial domain Ω_x -decomposition. The velocity grid size $N_v = 16^3$ and $N_v = 8^3$ are used for $Kn = 10, 1$ and $Kn = 0.1, 0.01$, respectively.

$Kn(Kn^* = 2.33Kn)$	0.01	0.1	1	10
MPI 1 × 1 × 1 OMP 12	10663 s (3700)	864.6 s (300)	7830 s (300)	13050 s (500)
MPI 2 × 2 × 2 OMP 12	1974 s (4100)	192.6 s (400)	4314 s (1100)	4706 s (1200)
MPI 4 × 4 × 4 OMP 12	377.4 s (5000)	45.28 s (600)	1397 s (2100)	1865 s (2800)
k	1.722×10^{-2}	3.862×10^{-2}	2.836×10^{-1}	3.375×10^0

N_v used for the smaller Kn (0.001 and 0.01) and larger Kn (0.1 and 1) cases are 8×8 and 24×24 , respectively. The Knudsen number Kn is defined with the reference length $L = 3000$ pixels, while the effective Knudsen number $Kn^* = 64.52Kn$ is estimated by Eq. (15) with $k_\infty = 1.201 \times 10^{-5}$. Similar to the previous Sierpinski carpet case, we show that the number of iterations to reach the converged permeabilities as presented in Table 2 and the convergence histories of permeability as shown in Fig. 3(b). The deterioration in convergence rate due to the spatial domain decomposition is found to be mitigated by increasing complexity in porous structure, especially at small Knudsen numbers. The increase of the total iterations is only 2% for $Kn = 0.001$ and 32% for $Kn = 1$, when 16 subdomains are used. The maximum deviation of the permeabilities predicted by the parallel solver from the corresponding serial ones is 0.024%. Comparing Fig. 3(b) with Fig. 3(a), we can find the convergence history of the random circular cylinders case is much smoother than the regular Sierpinski carpet case. This can be explained by the more pronounced mixing effect caused by the complex flow passages and gas-surface interactions.

4.2. 3D flows

Similar verifications are also conducted for our 3D parallel implementation of the algorithm on the flows through a simple

cubic array of spheres and the randomly packed spheres, which are often used as model porous media in analytical, computational and experimental studies [49,50].

In the first case, the unit cell is a cubic box with a sphere located at its center and repeated itself in the 3D space. Due to symmetry and periodicity of the configuration, we simulate only a quadrant of the unit cell as shown in Fig. 4(a) with the spatial grid points of $200 \times 100 \times 100$. The Knudsen number Kn is associated with the reference length $L = 200$ voxels, taken from the sample size in the x_1 direction, while the effective Knudsen number $Kn^* = 2.33Kn$ is estimated by Eq. (15) with $\epsilon = 0.75$ and $k_\infty = 1.722 \times 10^{-2}$. The number of iterations and the converged values of permeability are listed in Table 3, and the convergence histories are also presented in Fig. 4(c). The permeabilities predicted by the parallel solver deviate from the serial solver's results up to 0.11%. Similar to the 2D porous models, the degeneration of convergence rate also increases with number of spatial subdomains and Knudsen number. The number of iterations is increased by 35% and 460% for $Kn = 0.01$ and $Kn = 10$, respectively, when 64 subdomains are applied. We can also observe from Fig. 4(c) that the parallel solver's convergence history is not as smooth as the serial solver's at the high- Kn cases. One possible reason is that at a high Kn , distribution functions of some particles, which move freely between the inlet and the outlet, have been accidentally altered due to one-iteration-lag in information transfer at the boundaries of subdomain. These

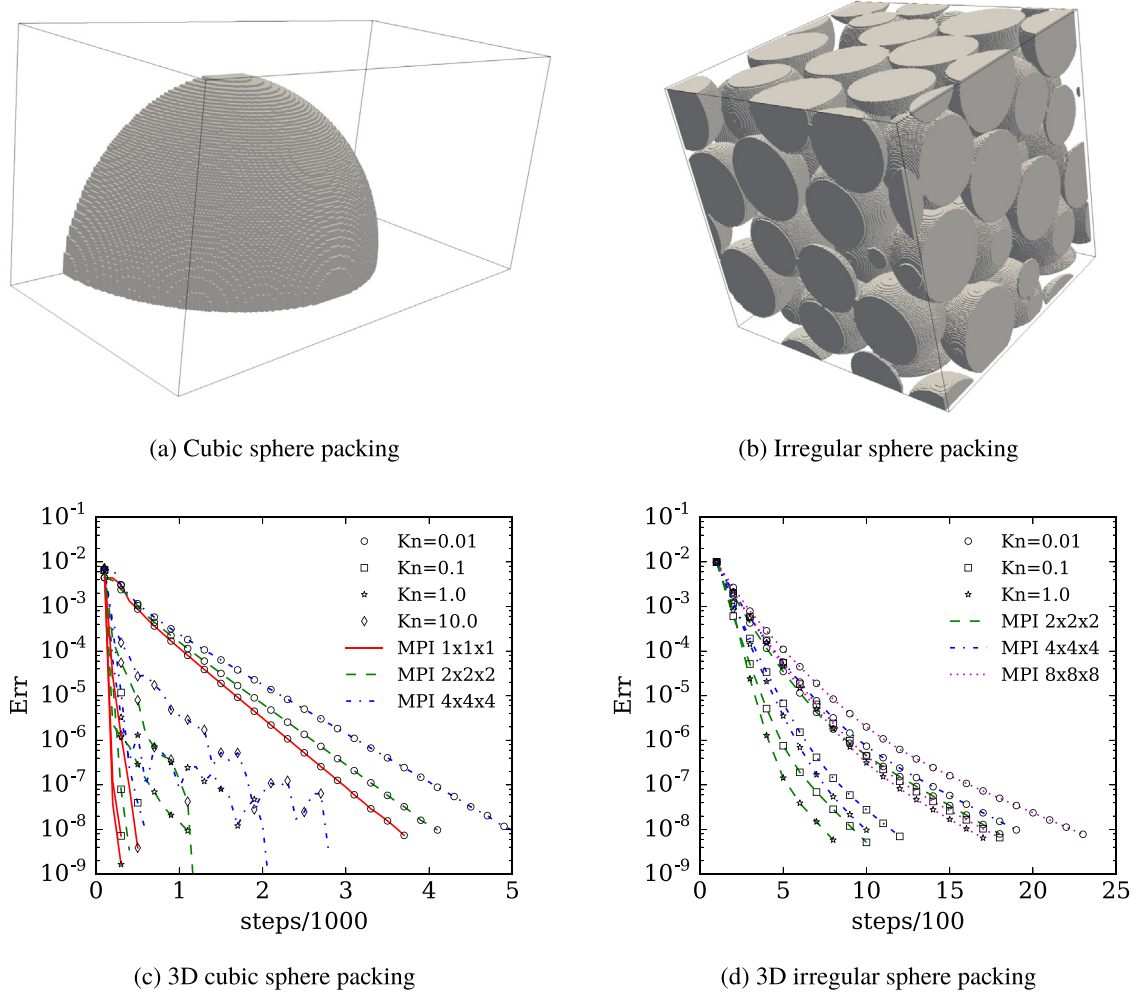


Fig. 4. The computational domains for 3D models of porous media made by sphere (in gray color) packing and the convergence history of the permeabilities given by Eq. (18) at different Knudsen numbers with different spatial domain decomposition. The spatial grid size N_x and porosity ϵ are (a) $N_x = 200 \times 100 \times 100$, $\epsilon = 0.75$, (b) $N_x = 308 \times 300 \times 300$, $\epsilon = 0.38$.

Table 4

The 3D porous model of irregular sphere packing [Fig. 4(b)]: the wall clock time (in second) and the number of iterations (in curve parentheses) at various Knudsen number Kn with different spatial domain Ω_x -decomposition. The data for pure OpenMP option is not available due to memory limit of one compute node.

Kn ($Kn^* = 19.4Kn$)	0.01	0.1	1
MPI $1 \times 1 \times 1$ OMP 12	n/a		
MPI $2 \times 2 \times 2$ OMP 12	8057 s (1800)	4475 s (1000)	3580 s (800)
MPI $4 \times 4 \times 4$ OMP 12	1318 s (1900)	831.8 s (1200)	692.9 s (1000)
MPI $8 \times 8 \times 8$ OMP 12	286.4 s (2300)	224.1 s (1800)	211.7 s (1700)
k	1.260×10^{-4}	7.823×10^{-4}	7.361×10^{-3}

alternations then assert their influence back to the inlet or outlet through periodic boundary conditions.

The second case is a cube filled with randomly packed spheres as illustrated in Fig. 4(b). Note that the inlet and outlet faces of a porous sample, in general, do not match exactly; therefore, additional fluid layers are needed to make periodic boundary conditions i.e. Eq. (10) applicable. We obtain fairly periodic density and velocity profiles at the inlet and outlet with extended four fluid layers at each face. The spatial grid size is $308 \times 300 \times 300$ including the extended fluid layers. The Knudsen number Kn is associated with the reference length $L = 308$ voxels, while the effective Knudsen number $Kn^* = 19.42Kn$ is estimated by Eq. (15) with $\epsilon = 0.38$ and $k_\infty = 1.260 \times 10^{-4}$. Kn of 0.01, 0.1 and 1 are

considered, and the velocity grid points of $8 \times 8 \times 8$ are deployed. The number of iterations together with the converged values of the permeabilities are listed in Table 4. From Fig. 4(d), it can be seen that the convergence history of the irregular sphere packing case is smooth even at high Kn , unlike the case of simple cubic array of spheres. Table 4 shows that the slowing down of convergence rate in the parallel solver is not as significant as the case of simple cubic array of spheres. The number of iterations increases by only 28% and 113% for $Kn = 0.01$ and $Kn = 1$, respectively, when the number of subdomains increases from 8 to 512. These observations in 3D porous media confirm our finding in the 2D cases that degeneration in convergence rate due to spatial domain decomposition is alleviated by complex porous media.

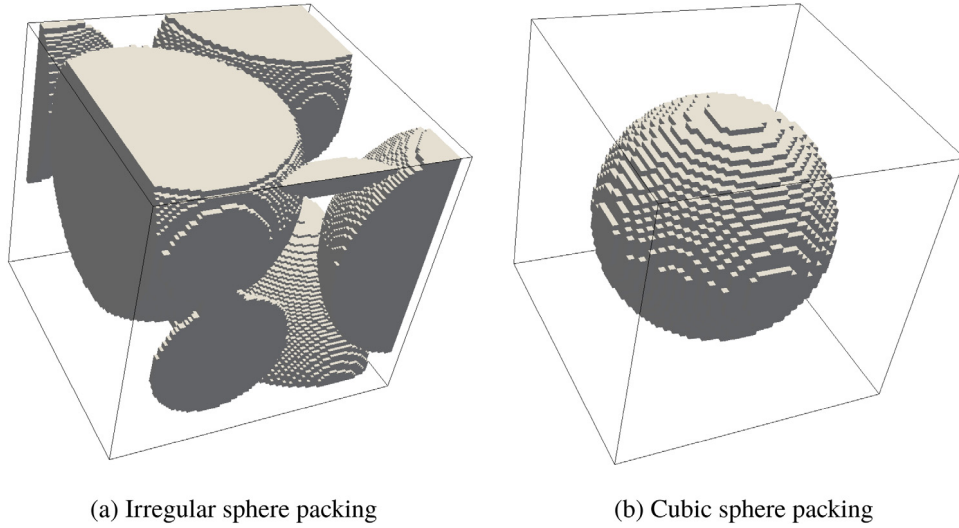


Fig. 5. Relatively small computation domains for 3D models of porous media made by sphere (in gray color) packing. The spatial grid size N_x and porosity ϵ are (a) $N_x = 108 \times 100 \times 100$, $\epsilon = 0.56$, (b) $N_x = 50 \times 50 \times 50$, $\epsilon = 0.73$.

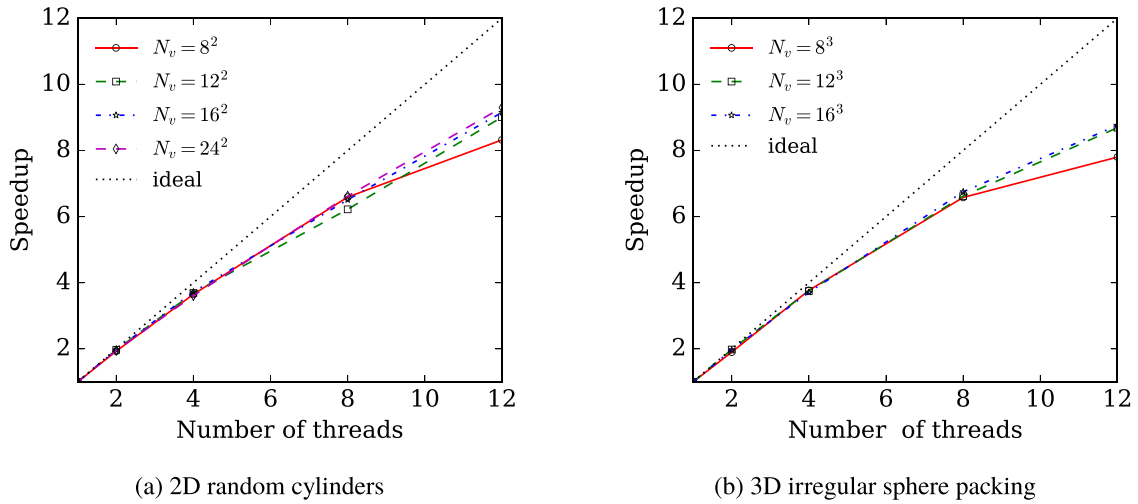


Fig. 6. Fine-grained (the bottom-level OpenMP) speedup for (a) the 2D porous model as shown in Fig. 2(b), and (b) the 3D porous model as shown in Fig. 5(a). Various velocity grid refinement N_v are considered.

5. Parallel performance

Measuring the scaling performance of the OpenMP and MPI parallelization individually allows us to better evaluate the overall efficiency of the multi-level parallel strategy. In this section, we demonstrate the scaling performance of each parallel level of the solver by excluding deterioration of convergence rate, i.e. comparing the wall clock time for an interval of 100 iterations. We first present the results of fine-grained (the bottom-level OpenMP) scaling performance using only one MPI process, followed by the weak and strong scaling performance with multiple MPI processes (the top level of parallelization), each with multiple OpenMP threads. Finally, speedup of two-level MPI/OpenMP parallelization is compared with that of pure MPI parallelization.

5.1. Fine-grained (the bottom-level OpenMP) scaling performance

To evaluate the OpenMP speedup against the serial processing, the OpenMP threads are mapped to the cores of a single processor (the maximum number of threads is 12) to avoid the performance drop caused by the non-uniform memory access (NUMA) across the two processors on the compute node.

The speedup of OpenMP parallelization for the 2D porous model as illustrated in Fig. 2(b) is plotted in Fig. 6(a). Almost perfect linear speedup is observed for all the examined velocity grids until 4 OpenMP threads. It can be seen that speedup slightly increases with the velocity grid refinement. In the case of 12 OpenMP threads, the speedup varies from 8.32 to 9.30 with the velocity grids of $N_v = 8^2$ and $N_v = 24^2$, respectively. The number of OpenMP threads can be augmented beyond the number of physical (12) cores by activating hyper-threading mode. However, the benefit of hyper-threading mode is only observed in the case of the smallest velocity grid $N_v = 8^2$. In particular, the maximum speedup without hyper-threading (12 OpenMP threads) is 8.3, 9.1, 9.3 and with hyper-threading on (24 OpenMP threads) is 10.4, 8.0, 7.7 for $N_v = 8^2, 16^2, 24^2$, respectively.

Fig. 6(b) plots the speedup of OpenMP parallelization for the 3D porous model as shown in Fig. 5(a), which is a portion extracted from irregular sphere packing as shown in Fig. 4(b). Similar to the 2D case, speedup is almost linear until 4 OpenMP threads. In the case of 12 OpenMP threads, the speedup increases slightly from 7.80 to 8.76 with the velocity grid refinement from $N_v = 8^3$ to $N_v = 16^3$. This can be attributed to the better load balancing

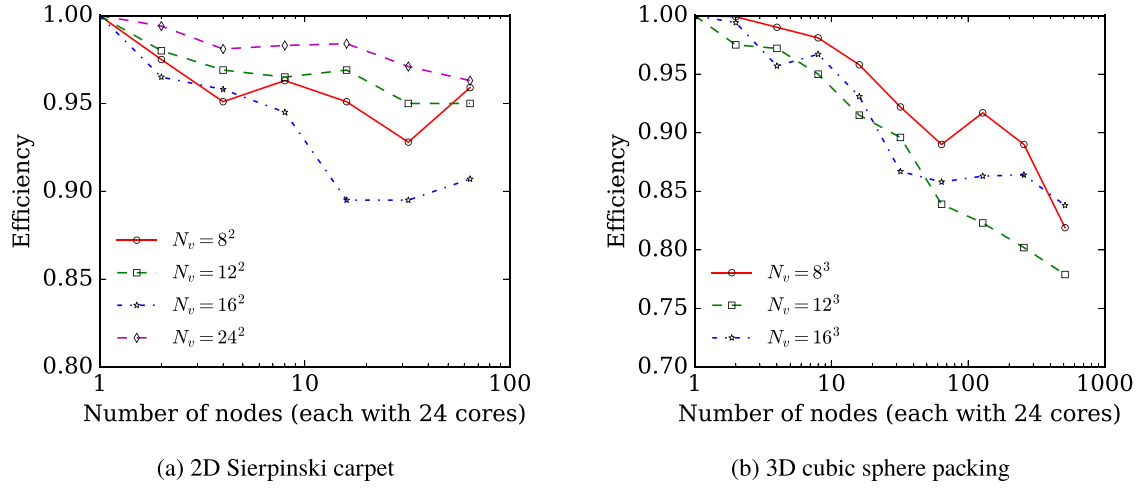


Fig. 7. The coarse-grained (the top-level MPI) efficiency of weak scaling, in which each MPI process handles a subdomain (a) as shown in Fig. 2(a) but with a coarser resolution $N_x = 270 \times 270$; and (b) as shown in Fig. 5(b). Various velocity grid refinement N_v are considered.

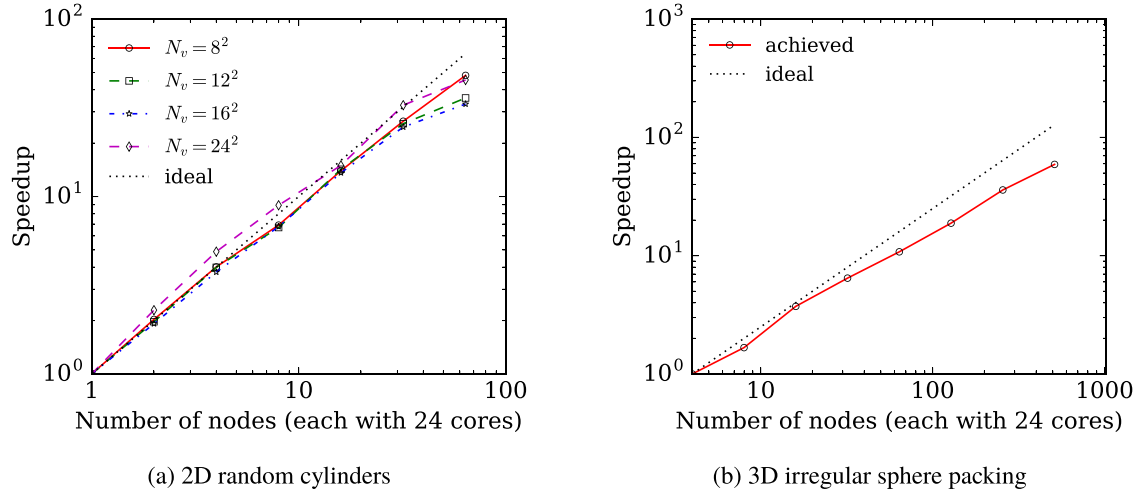


Fig. 8. The coarse-grained (the top-level MPI) speedup of strong scaling for (a) the 2D porous model as shown in Fig. 2(b); and (b) the 3D porous model as shown in Fig. 4(b). Velocity grid of $N_v = 8^2, 12^2, 16^2, 24^2$ are considered in (a), and $N_v = 8^3$ in (b).

among the OpenMP threads when the number of discrete velocity increases, as the per thread work chunk size increases accordingly.

5.2. Coarse-grained (the top-level MPI) scaling performance

In order to measure the MPI scalability, the number of OpenMP threads for each MPI process is fixed at 12, and each compute node is assigned to 2 MPI processes, running on the two processors. All MPI scalabilities are assessed against 1 compute node apart from the case of 3D strong scaling, where 4 compute nodes are used as the baseline due to memory requirement.

Weak scaling examines parallel performance by assuming that the spatial domain increases at the same rate as the number of cores, i.e. the workload per subdomain is equally fixed. Fig. 7(a) shows the weak scaling efficiency of the 2D solver, in which each MPI process manages a spatial subdomain as illustrated in Fig. 2(a) with a coarse resolution of $N_x = 270 \times 270$ pixels. The weak scaling efficiency only slightly decreases with increasing number of MPI processes. Very good parallel efficiency of about 94% is observed with a total of 128 MPI processes (1536 cores). In Fig. 7(b), similar observation is found for the 3D solver, in which each MPI process handles a spatial subdomain as shown in Fig. 5(b). The weak scaling efficiency gradually declines to 86% for the total of

128 MPI processes (1536 cores) and 81% for the 1024 MPI processes (12288 cores). It is worth noting that speedup of 420 is obtained with 512 compute nodes for the grid size of $1.1 \times 10^9 \cdot 4.1 \times 10^3$ as examined on this weak scaling. High efficiency of the weak scaling indicates the solvers' capability for a large number of voxels of homogeneous porous media.

On the other hand, strong scaling examines the parallel performance when the computational domain remains unchanged as the number of cores increases. Fig. 8(a) and (b) demonstrate the strong scaling speedup of the 2D and 3D solvers on the spatial domains illustrated in Figs. 2(b) and 4(b), respectively. It can be seen that the 2D strong scaling speedup is close to ideal linear one until 64 MPI processes (768 cores), where the efficiency is around 86%, and the efficiency falls to 64% at 128 MPI processes (1536 cores). The super linear speedup for $N_v = 24^2$ observed in Fig. 8(a) may be due to the cache miss, i.e. when the processor fails to request data in the cache memory, the processor has to wait for the data to be fetched from the higher level cache or from the main memory with much longer latency. The larger problem size, the more chance to encounter cache miss as larger requested data may not fit into cache size. We perform additional strong scaling test for an extracted part (2000×1000 pixels) of the original 2D sample (3000×1500 pixels). Therefore, the domain size in this

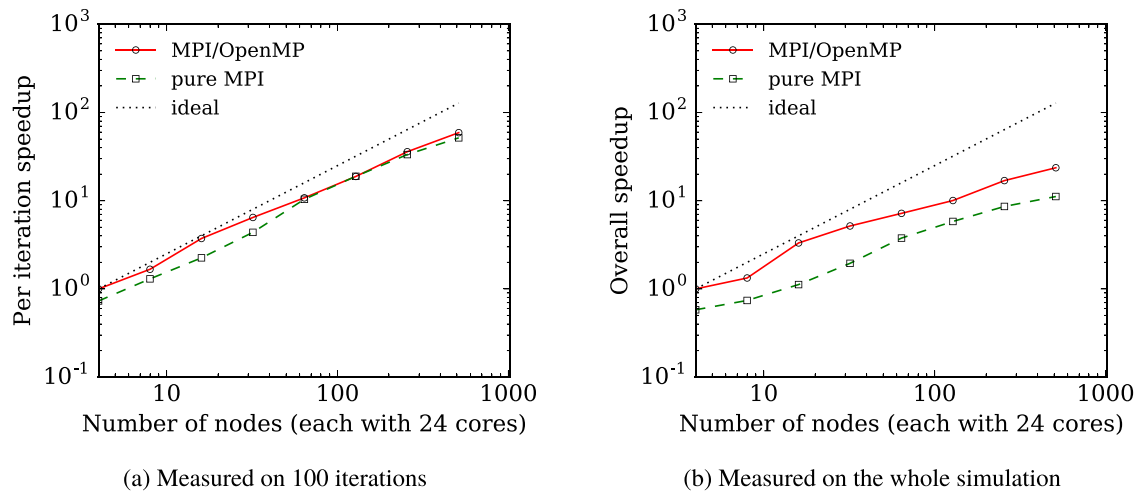


Fig. 9. Parallel speedup of strong scaling for the 3D irregular sphere packing as shown in Fig. 4(b): multi-level MPI/OpenMP mode versus pure MPI mode. The velocity grid of $N_v = 8^3$ is used for the case of $Kn = 1$.

test with $N_v = 24^2$ is similar to that in the original case with $N_v = 16^2$, where the super linear speedup is not observed. The profiling results show that the super linear phenomenon indeed does not appear again, and the trend of the speedup with the number of MPI processes is now similar to the other cases of $N_v = 8^2, 12^2, 16^2$. In the 3D case, strong scaling speedup keeps increasing even beyond 12288 cores, but the efficiency slumps from 81% to 67% as the number of MPI processes increases from 64 to 128 (768 to 1536 cores). The deterioration in strong scaling performances can be explained by load imbalance among the MPI processes, which is directly related to inhomogeneity of the porous models. Consider the cases of 128 MPI processes, where the spatial domains are decomposed into 16×8 rectangular subdomains for the 2D random cylinders as shown in Fig. 2(b) or $4 \times 4 \times 8$ cuboid subdomains for the 3D irregular sphere packing as shown in Fig. 4(b). The porosity ϵ of each subdomain varies from 0.313 to 0.909 for the 2D case ($\epsilon = 0.60$ on average) or from 0.090 to 0.706 for the 3D case ($\epsilon = 0.38$ on average), leading to strong load imbalance among the MPI processes. Improving load balance for heterogeneous porous media is certainly worth further studies, e.g. using the standard graph partitioners such as METIS [51], PT-SCOTCH [52] or developing suitable algorithms.

5.3. Scaling performance of multi-level MPI/OpenMP versus pure MPI parallelism

As mentioned in Section 1, pure MPI parallelization commonly employs either velocity or spatial domain decomposition. But the velocity domain decomposition is not suitable for a relatively large spatial domain. It is interesting to compare the parallel performance of multi-level MPI/OpenMP approach adopted in this study with that of pure MPI approach using spatial domain decomposition. To enable pure MPI mode from the adopted multi-level parallel solvers, we disable OpenMP recognition by adding the compiler flag `-hnoomp` in the Cray Programming Environment. The wall clock time of both 100 iterations and whole simulations obtained by the multi-level and pure MPI modes for the case of 3D irregular sphere packing as shown in Fig. 4(b) with $Kn = 1$ is recorded. The wall clock time of the multi-level mode on 4 compute nodes is used as the baseline time for calculating speedup. Fig. 9(a), in which the speedup on the interval of 100 iterations is reported, shows a relatively good scaling of both parallel modes until the maximum used resource, i.e. 12288 cores. With the same resource, the multi-level mode is between 30% and 60% faster than the pure MPI mode until 768 cores and becomes slightly faster beyond

that number of cores. From Fig. 9(b), when the deterioration of convergence rate is taken into account, it can be observed that even at 512 compute nodes (12288 cores) the overall speedup of both the multi-level and pure MPI modes have not been saturated. However, with the same number of CPU cores, the overall speedup of multi-level mode is roughly 1.5 to 3 times better than that of the pure MPI mode. One main reason is that the multi-level mode has less number of spatial subdomains, i.e. MPI processes, than the pure MPI mode, resulting in faster convergence rate for the iterative scheme.

6. Conclusions and remarks

In this work, we have developed a high-performance DVM solver for steady 2D and 3D rarefied gas flows in porous media.

While spatial domain decomposition is inevitable for practical large-scale pore-scale computations, the number of subdomains should be kept minimal to avoid deterioration of the convergence rate of the iterative scheme. Therefore, two-level MPI/OpenMP parallelization is proposed to improve parallel performance, where an additional parallel level (OpenMP) allows further speedup with the fixed number of subdomains. In addition, it mitigates deterioration in convergence rate with the fixed CPU resource. The parallel scaling shows that the two-level parallel approach has significantly better performance than the commonly-used MPI approach for the same number of CPU cores. The deterioration of the convergence rate is found to become worse with increasing Knudsen number, but it can be mitigated by complex porous media.

The developed solver can enable 3D pore-scale simulations to predict the flow properties of porous media. Further investigations on optimization of grid partition is needed to improve scalability for heterogeneous porous media. This multi-level parallel approach, which is demonstrated with the BGK equation here, can be easily extended to solve other kinetic model equations. With recent releases of OpenMP supporting for GPUs, the proposed solver may be readily adapted for hybrid CPU/GPU clusters.

Acknowledgments

Financial support from the UK Engineering and Physical Sciences Research Council (EPSRC) under grant no. EP/M021475/1 is gratefully acknowledged. L. Zhu acknowledges the financial support from the Chinese Scholarship Council (CSC) during his visit to the UK (CSC Student no. 201606160050). L. Wu acknowledges the financial support from the RSE-NSFC, China joint project

and Carnegie Research Incentive Grant, United Kingdom. Computing time on the ARCHER is provided by the “UK Consortium on Mesoscale Engineering Sciences (UKCOMES)” under the UK EPSRC grant no. EP/L00030X/1.

References

- [1] Q. Wang, X. Chen, A.N. Jha, H. Rogers, *Renewable Sustainable Energy Rev.* 30 (2014) 1–28, <http://dx.doi.org/10.1016/j.rser.2013.08.065>.
- [2] E. Harel, J. Granwehr, J.A. Seeley, A. Pines, *Nature Mater.* 5 (4) (2006) 321–327, <http://dx.doi.org/10.1038/nmat1598>.
- [3] H. Darabi, A. Ettehad, F. Javadpour, K. Sepehrnoori, *J. Fluid Mech.* 710 (2012) 641–658, <http://dx.doi.org/10.1017/jfm.2012.424>.
- [4] I. Lunati, S.H. Lee, *J. Fluid Mech.* 757 (2014) 943–971, <http://dx.doi.org/10.1017/jfm.2014.519>.
- [5] J.C. Maxwell, *Philos. Trans. R. Soc. Lond.* 170 (1879) 231–256, <http://dx.doi.org/10.1098/rstl.1879.0067>.
- [6] F. Sharipov, *J. Phys. Chem. Ref. Data* 40 (2) (2011) 023101, <http://dx.doi.org/10.1063/1.3580290>.
- [7] D.A. Lockerby, J.M. Reese, *J. Fluid Mech.* 604 (2008) 235–261, <http://dx.doi.org/10.1017/S0022112008001158>.
- [8] G.A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford Science Publications, 1994, p. 459.
- [9] L. Pareschi, B. Perthame, *Transport Theory Statist. Phys.* 25 (3–5) (1996) 369–382, <http://dx.doi.org/10.1080/00411459608220707>.
- [10] V.I. Kolobov, R.R. Arslanbekov, V.V. Aristov, A.A. Frolova, S.A. Zabelok, *J. Comput. Phys.* 223 (2) (2007) 589–608, <http://dx.doi.org/10.1016/j.jcp.2006.09.021>.
- [11] Y.Y. Kloss, F.G. Cheremisin, N.I. Khokhlov, B.A. Shurygin, *At. Energy* 105 (4) (2008) 270–279, <http://dx.doi.org/10.1007/s10512-009-9096-3>.
- [12] T. Scanlon, E. Roohi, C. White, M. Darbandi, J. Reese, *Comput. & Fluids* 39 (10) (2010) 2078–2089, <http://dx.doi.org/10.1016/j.compfluid.2010.07.014>.
- [13] A. Frezzotti, G.P. Ghiroldi, L. Gibelli, *Comput. Phys. Comm.* 182 (12) (2011) 2445–2453, <http://dx.doi.org/10.1016/j.cpc.2011.07.002>.
- [14] L. Wu, C. White, T.J. Scanlon, J.M. Reese, Y. Zhang, *J. Comput. Phys.* 250 (2013) 27–52, <http://dx.doi.org/10.1016/j.jcp.2013.05.003>.
- [15] P. Bhatnagar, E. Gross, M. Krook, *Phys. Rev.* 94 (3) (1954) 511–525, <http://dx.doi.org/10.1103/PhysRev.94.511>.
- [16] L.H. Holway, *Phys. Fluids* 9 (9) (1966) 1658–1673, <http://dx.doi.org/10.1063/1.1761920>.
- [17] E.M. Shakhov, *Fluid Dyn.* 3 (5) (1968) 95–96, <http://dx.doi.org/10.1007/BF01029546>.
- [18] Y.H. Qian, D. D’Humières, P. Lallemand, *Europhys. Lett.* 17 (6) (1992) 479, <http://dx.doi.org/10.1209/0295-5075/17/6/001>.
- [19] M.J. Blunt, B. Bijeljic, H. Dong, O. Gharbi, S. Iglauer, P. Mostaghimi, A. Paluszny, C. Pentland, *Adv. Water Resour.* 51 (2013) 197–216, <http://dx.doi.org/10.1016/j.advwatres.2012.03.003>.
- [20] J. Wang, L. Chen, Q. Kang, S.S. Rahman, *Int. J. Heat Mass Transfer* 95 (2016) 94–108, <http://dx.doi.org/10.1016/j.ijheatmasstransfer.2015.12.009>.
- [21] S.H. Kim, H. Pitsch, I.D. Boyd, *J. Comput. Phys.* 227 (19) (2008) 8655–8671, <http://dx.doi.org/10.1016/j.jcp.2008.06.012>.
- [22] J. Meng, Y. Zhang, *J. Comput. Phys.* 230 (3) (2011) 835–849, <http://dx.doi.org/10.1016/j.jcp.2010.10.023>, [arXiv:0908.4520v2](https://arxiv.org/abs/0908.4520v2).
- [23] J. Meng, Y. Zhang, *Phys. Rev. E* 83 (3) (2011) 36704, <http://dx.doi.org/10.1103/PhysRevE.83.036704>.
- [24] J. Meng, Y. Zhang, N.G. Hadjiconstantinou, G.A. Radtke, X. Shan, *J. Fluid Mech.* 718 (2013) 347–370, <http://dx.doi.org/10.1017/jfm.2012.616>.
- [25] A. Frezzotti, G.P. Ghiroldi, L. Gibelli, *Comput. & Fluids* 50 (1) (2011) 136–146, <http://dx.doi.org/10.1016/j.compfluid.2011.07.004>.
- [26] G. Dimarco, R. Loubère, J. Narski, *J. Comput. Phys.* 284 (2015) 22–39, <http://dx.doi.org/10.1016/j.jcp.2014.12.023>.
- [27] V.A. Titarev, *Commun. Comput. Phys.* 12 (1) (2012) 162–192, <http://dx.doi.org/10.4208/cicp.220111.140711a>.
- [28] V. Titarev, M. Dumbser, S. Utyuzhnikov, *J. Comput. Phys.* 256 (2014) 17–33, <http://dx.doi.org/10.1016/j.jcp.2013.08.051>.
- [29] V.A. Titarev, S.V. Utyuzhnikov, A.V. Chikritkin, *Comput. Math. Math. Phys.* 56 (11) (2016) 1919–1928, <http://dx.doi.org/10.1134/S0965542516110129>.
- [30] Z.H. Li, H.-X. Zhang, *J. Comput. Phys.* 228 (4) (2009) 1116–1138, <http://dx.doi.org/10.1016/j.jcp.2008.10.013>.
- [31] Z.H. Li, A.P. Peng, H.-X. Zhang, J.-Y. Yang, *Prog. Aerosp. Sci.* 74 (2015) 81–113, <http://dx.doi.org/10.1016/j.paerosci.2014.12.002>.
- [32] C. Baranger, J. Claudel, N. Hérouard, L. Mieussens, *J. Comput. Phys.* 257 (2014) 572–593, <http://dx.doi.org/10.1016/j.jcp.2013.10.014>.
- [33] L. Zhu, S. Chen, Z. Guo, *Comput. Phys. Comm.* 213 (2017) 155–164, <http://dx.doi.org/10.1016/j.cpc.2016.11.010>.
- [34] R. Rabenseifner, G. Hager, G. Jost, 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 2009, pp. 427–436, <http://dx.doi.org/10.1109/PDP.2009.43>.
- [35] J.Y. Yang, J.-C. Huang, *J. Comput. Phys.* 120 (2) (1995) 323–339, <http://dx.doi.org/10.1006/jcp.1995.1168>.
- [36] L. Mieussens, *J. Comput. Phys.* 162 (2) (2000) 429–466, <http://dx.doi.org/10.1006/jcp.2000.6548>.
- [37] K. Xu, J.-C. Huang, *J. Comput. Phys.* 229 (20) (2010) 7747–7764, <http://dx.doi.org/10.1016/j.jcp.2010.06.032>.
- [38] F. Filbet, S. Jin, *J. Comput. Phys.* 229 (20) (2010) 7625–7648, <http://dx.doi.org/10.1016/j.jcp.2010.06.017>.
- [39] Z. Guo, K. Xu, R. Wang, *Phys. Rev. E* 88 (3) (2013) 33305, <http://dx.doi.org/10.1103/PhysRevE.88.033305>.
- [40] Y. Zhu, C. Zhong, K. Xu, *J. Comput. Phys.* 315 (2016) 16–38, <http://dx.doi.org/10.1016/j.jcp.2016.03.038>.
- [41] D. Sharov, H. Luo, J. Baum, R. Loehner, 38th Aerospace Sciences Meeting and Exhibit, in: Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics, 2000, <http://dx.doi.org/10.2514/6.2000-927>.
- [42] F. Sharipov, V. Seleznev, *J. Phys. Chem. Ref. Data* 27 (3) (1998) 657–706, <http://dx.doi.org/10.1063/1.556019>.
- [43] F. Sharipov, I.A. Graur, *Vacuum* 86 (11) (2012) 1778–1782, <http://dx.doi.org/10.1016/j.vacuum.2012.02.028>.
- [44] J.E. Broadwell, *J. Fluid Mech.* 19 (03) (1964) 401–414, <http://dx.doi.org/10.1017/S0022112064000817>.
- [45] W. Gautschi, *ACM Trans. Math. Software* 20 (1) (1994) 21–62, <http://dx.doi.org/10.1145/292395.292467>, [arxiv:9307212](https://arxiv.org/abs/9307212).
- [46] L. Wu, M.T. Ho, L. Germanou, X.J. Gu, C. Liu, K. Xu, Y. Zhang, *J. Fluid Mech.* 822 (2017) 398–417, <http://dx.doi.org/10.1017/jfm.2017.300>.
- [47] D. Valougeorgis, S. Naris, *SIAM J. Sci. Comput.* 25 (2) (2003) 534–552, <http://dx.doi.org/10.1137/S1064827502406506>.
- [48] P. Wang, M.T. Ho, L. Wu, Z. Guo, Y. Zhang, *Comput. & Fluids* 161 (2018) 33–46, <http://dx.doi.org/10.1016/j.compfluid.2017.11.006>, [arXiv:1612.06590](https://arxiv.org/abs/1612.06590).
- [49] A. Sangani, A. Acrivos, *Int. J. Multiph. Flow* 8 (4) (1982) 343–360, [http://dx.doi.org/10.1016/0301-9322\(82\)90047-7](http://dx.doi.org/10.1016/0301-9322(82)90047-7).
- [50] R.J. Hill, D.L. Koch, A.J.C. Ladd, *J. Fluid Mech.* 448 (2001) 243–278, <http://dx.doi.org/10.1017/S0022112001005936>.
- [51] G. Karypis, V. Kumar, *SIAM J. Sci. Comput.* 20 (1) (1998) 359–392, <http://dx.doi.org/10.1137/S1064827595287997>.
- [52] C. Chevalier, F. Pellegrini, *Parallel Comput.* 34 (6–8) (2008) 318–331, <http://dx.doi.org/10.1016/j.parco.2007.12.001>.