International Journal of Heat and Mass Transfer 115 (2017) 1192–1202

Contents lists available at ScienceDirect

International Journal of Heat and Mass Transfer

journal homepage: www.elsevier.com/locate/ijhmt

Unified mesoscopic modeling and GPU-accelerated computational method for image-based pore-scale porous media flows

Senyou An [a,b], Huidan (Whitney) Yu [a,*], Zhiqiang Wang [c], Behram Kapadia [a], Jun Yao [b,*]

[a] Department of Mechanical Engineering, Indiana University-Purdue University, Indianapolis, IN 46202, USA
[b] School of Petroleum Engineering, China University of Petroleum, Qingdao 266580, China
[c] Department of Computer Science, Kent State University, Kent, OH 44242, USA

ARTICLE INFO

Article history:
Received 17 April 2017
Received in revised form 26 August 2017
Accepted 28 August 2017
Available online 8 September 2017

Keywords:
Pore-scale porous media flow
Lattice Boltzmann method
GPU parallelism
Image segmentation
Pore-structure upscaling

ABSTRACT

Solving Navier-Stokes equations for pore-scale porous media flows (PSPMFs) based on imaging data has attracted increasing attention in the community of porous media flows due to a variety of application needs. However, the existing numerical approach is laborious, complex, and error-prone utilizing an ad-hoc coalition of software packages. The multidisciplinary tasks of image processing, computational fluid modeling, and high performance computing together with their integration is overwhelming. We present a unique and powerful computational platform, for PSPMFs with physical and computational advantages. This systematic method is featured with (1) unified mesoscopic modeling, (2) GPU (Graphic Processing Units) parallel computing, and (3) pore-structure upscaling. The multidisciplinary tasks are innovatively integrated into one computational setup thus no software coalition is needed. We use lattice Boltzmann method (LBM) to solve the level-set equation and NS equations successively, thus image segmentation and computational fluid dynamics are seamlessly connected avoiding extra grid and mesh generation or data transfer. The unified LBM modeling enables high efficient GPU parallelism for scalable acceleration. Meanwhile, a new pore-structure upscaling scheme is developed and integrated into downscale the fine grid mesh in the premise of ensuring accuracy. Two application studies demonstrate the reliability, feasibility, and efficiency of this method for PSPMFs. It is believed to be the first-ever integrated and accelerated computational package for studying PSPMFs and it is sustainable. We will introduce more physical models such as interfacial dynamics for multiphase flows, fluid–structure interaction for deformable pore structure, non-Newtonian effects, etc. in the near future. Meanwhile, parallelism based on multiple GPU cards for further acceleration is being investigated.

1. Introduction

Porous media flows are ubiquitous in nature and engineering applications. A few examples include the propagation of chemical contaminants in underground hydrology field, ink permeation, sedimentation, storage of hazardous wastes, flow in oil reservoirs, and biological tissues. Conventionally, due to the lack of appropriate research tools, porous media flow meant spatially and temporally averaged flow solved by phenomenologically and empirically derived constitutive equations such as Darcy's law [1]. In recent years, pore-scale porous media flow (PSPMF) has attracted increasing attentions by utilizing newly developed radiological imaging techniques such as computed tomography (CT), nuclear magnetic resonance spectroscopy (NMR), and magnetic resonance imaging (MRI); thus the celebrated Navier-Stokes (NS) equations have become the governing equations for porous media flows in pore space. Experimental investigation of PSPMF is inherently difficult due to the lack of optical access, whereas computational fluid dynamics (CFD) is a promising alternative to be capable of solving the complete fluid dynamics taking into account of heterogeneity, complex pore inter-connectivity, and morphologies of porous media [2,3]. In addition to the capability to access flows in pore space, CFD has attractive advantages including low cost of facility, personnel, and supplies, short and flexible time cycle, and easy setup to mimic real-world flow conditions.

1.1. Existing CFD approach for pore-scale porous media flow

Existing CFD approaches for solving PSPMF based on imaging data is laborious, complex, and error-prone, as schematized in Fig. 1. From imaging data to numerical fluid dynamics, multidisciplinary tasks consist of 3-D reconstruction to extract pore-structure from images, mesh generation to connect the segmented

* Corresponding authors.
E-mail addresses: whyu@iupui.edu (H. (Whitney) Yu), yaojun@upc.edu.cn (J. Yao).

http://dx.doi.org/10.1016/j.ijheatmasstransfer.2017.08.099
0017-9310/

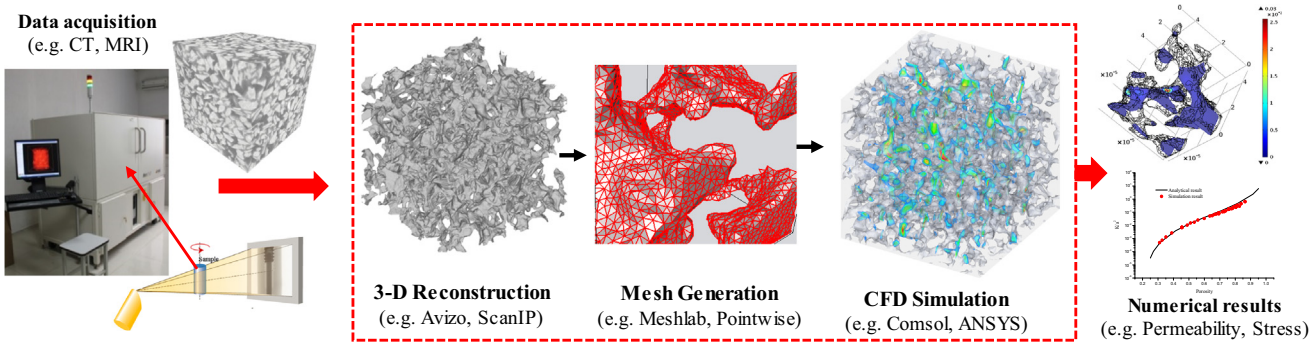

Fig. 1. Schematic of prevalent computational approach for solving PSPMF based on imaging data.

image data to a CFD solver, and CFD simulation to solve the complete fluid dynamics in pore space. These tasks, each of which is tedious and requires a specific disciplinary training, are currently accomplished through an ad-hoc coalition of software packages (a few examples are shown in the red box). Such an approach requires proficient skills in four disciplines. (1) *Sufficient training for CFD:* Most existing CFD software consist of multiple but separate modules thus requiring sophisticated CFD training for design of complex 3-D meshes, selection of computational models and physical attributes, and implementation of high-performance computing. (2) *Advanced level training for nondestructive image processing:* Radiological images inherently contain artifacts. Obscuration of significant features or misinterpretation of attenuation values of a single material in different image sections may significantly complicate quantitative image analysis [4]. Commonly experienced problems of X-ray images include high-frequency noise, beam hardening, defective detector pixels, scattered X-rays or poorly centered samples[5]. Therefore, being able to nondestructively extract pore structure in a large 3-D volume under given spatial resolution is crucial. (3) *Hands-on experience to connect segmented pore structure to CFD software:* each software has its own data format for inputs and outputs. From an image segmentation software to a CFD software, it is required to deal with the 3-D geometry formation and mesh generation by utilizing additional software. This requirement demands adequate hands-on experience operating interdisciplinary software packages. (4) Capability of *high-performance computation associated with massive data handling*: the computational cost for a PSPMF is inherently significant based on porosity and pore size distribution. For example, an underground rock core usually contains pore scales across 2–3 decades, e.g. from 0.2 to 100 μm. If a high resolution Micro-CT with pixel size 5 μm [6] is used to scan such a rock sample, a 0.5 cm^3 sample may result in a 1000^3 grid size, demanding very expensive computation. Powerful computer resources (e.g. remote access high-performance computing facility or supercomputers) are necessary to meet the high computation cost. Therefore, sufficient skills in parallel computation and large data handling are essential. Overall, the multidisciplinary nature of the existing approach makes it overwhelming to study PSPMFs. It is very important to develop new computational methodology with integrated-modeling and efficient computation.

1.2. Macroscopic vs. mesoscopic modeling for porous media flows

Macroscopic modeling is dominated in current CFD software packages, which solves NS equations using Finite Element Method in COMOSOL and Finite Volume Method in OpenFOAM and Fluent for fluid dynamics. Modeling on this level has its inherent difficulties for dealing with flows in 3-D arbitrary geometry such as porous media. Existing software packages usually use unstructured grids for adaptive subdivision for 3-D boundaries. As a result, bad elements usually appear in complex pore structure, e.g. the tips and small isolated space in digital core. PSPMF can be much more easily modeled on microscopic modeling such as Molecular Dynamics [7] but the computation cost is too high to meet the needs of most applications. In between, particle-based macroscopic approach like smoothed particle hydrodynamics (SPH) [8] or mesoscopic fluid solvers such as dissipative particle dynamics [9] and lattice Boltzmann method (LBM) [10] have been known more suitable for PSPMFs. Among those, SPH and LBM have been widely employed, but SPH still needs further development on its theoretical foundations [8]. Challenges exist in SPH to reproduce the governing PDEs (partial differential equations) for the fluid and solid mechanics [8]. Meanwhile, solving pore scale flow needs very fine resolution thus fast computing demands. Whereas the LBM has been proved well amenable for PSPMFs because it is not only easy to handle complex and arbitrary pore structure [11–13], but also ideally suited for high efficient GPU (Graphic Processing Units) parallelization [14].

1.3. Lattice Boltzmann method for pore-scale porous media flows

The LBM deals with the time evolution of particle density distribution function associated with a prescribed finite discrete set of particle velocities on a lattice space [10]. Fluid particles are uniformly distributed at lattice nodes. They collide at the nodes and then stream to the neighboring nodes along the defined directions. The fluid dynamic variables are obtained via moments of the particle distribution functions. Through the Chapman-Enskog technique, it has been rigorously proven [15] that the lattice Boltzmann equations recover NS equations to the second-order accuracy in both space and time. There are three attractive advantages of LBM for porous media flows. The first is the *ease of handling the complicated geometry* of pores. The no-slip bounce-back boundary condition [16] on local pore structure costs little implementation effort [17] and computational time [18]. The second is the *amenability of scalable parallelization* over fine-grained parallel architectures such as GPU accelerators [19]. The third is the *suitability of introducing intermolecular interactions* for multiphase flows [20] and recovering the appropriate multiphase dynamics without a demanding computation cost [11]. Meanwhile, the LBM has been extended to deal with image de-noising and segmentation [21]. The inherent parallel data structure facilitates fast surface reconstruction from 3-D radiological images with large time steps and satisfied accuracy. Unified LBM modeling make it possible to connect image segmentation and CFD seamlessly and sufficiently utilize GPU acceleration.

In this work, we present a novel computational method, to solve PSPMFs based on imaging data. This method is featured with (1) unified mesoscopic modeling, (2) GPU parallel computing, and

(3) pore-structure upscaling. The multidisciplinary tasks mentioned in Section 1.1 are innovatively integrated into one computational setup thus no software coalition is needed. We use LBM to solve the level-set equation and NS equations successively, thus image segmentation and CFD are seamlessly connected avoiding extra grid and mesh generation or data transfer. The unified LBM modeling enables high efficient GPU parallelism for scalable acceleration. Meanwhile, a new pore-structure upscaling scheme is developed and integrated into downscale the fine grid mesh in the premise of ensuring accuracy. Two application studies demonstrate the reliability, feasibility, and efficiency of this computational approach for PSPMFs.

The remainder of the paper is organized as follows. In Section 2, we introduce the computational methodology including unified LBM modeling, GPU parallelism, and pore-structure upscaling scheme. Applications studies are presented in Section 3. Finally, Section 4 provides a summary discussion and concludes the paper.

2. Computational methodology

This systematic computational platform integrate mesoscopic modeling with GPU parallelism for solving image-based PSPMFs. For the purpose of expanding the capability of this method to deal with realistic porous media systems, in the order of cm^3 comparable to the system size studied in laboratory experiment, pore-structure upscaling scheme is developed and integrated.

2.1. Unified LBM modeling for image segmentation and fluid dynamics

First, the pore structure is extracted on a uniform lattice mesh from provided image data with certain resolution. The pore structure, expressed by the volumetric ratio of solid versus fluid of each lattice cell, is then fed to the next step for solving fluid dynamics using volumetric LBM. The **KEY** of seamless connection is the calculation of solid volumetric ratio in each cell after the image segmentation.

2.1.1. Lattice Boltzmann method for 3-D image segmentation

Level-set method (LSM) [22] has been one of the most important tools for image segmentation. It uses a PDE to model and track evolving fronts in a discrete domain by maintaining and updating a distance field to the fronts. The distance field implicitly represents the geometric contours or surfaces when the fronts converge at the boundary of the aimed object. Let $I_0(\boldsymbol{x},t)$ be the original 3-D volume data of images and denote Γ the evolving surface with the inward normal direction $\boldsymbol{N}_{\text{in}}$. Define a signed distance field $\phi(\boldsymbol{x},t)$ based on the surface Γ (as shown in Fig. 2): $\phi(\boldsymbol{x},t)<0$ if $\phi(\boldsymbol{x},t)\in\Gamma$, $\phi(\boldsymbol{x},t)>0$ if $\phi(\boldsymbol{x},t)\notin\Gamma$, and $\phi(\boldsymbol{x},t)=0$ if $\phi(\boldsymbol{x},t)$ is on Γ. Meanwhile, $|\nabla\phi|=1$ for a distance field. A level set equation models the time evolution of the distance field $\phi(\boldsymbol{x},t)$ as

$$\partial\phi/\partial t=\nabla\cdot(g\nabla\phi)+\beta g \tag{1}$$

where β is an adjustable constant and g is an edge-stopping function [22] defined by $g=1/(1+|\nabla G\times I_0|^2)$ [23] where the term $|\nabla G\times I_0|$ denotes the gradient of a convolution between the original image $I_0(\boldsymbol{x})$ and a Gaussian kernel $G(\boldsymbol{x},t)$. A level set solver converges when $\phi(\boldsymbol{x},t)$ is no longer evolving according to g defined on $I_0(\boldsymbol{x},t)$ based on image attributes. Then Γ represents the boundary of the preferred object over $I_0(\boldsymbol{x},t)$. Traditionally, the level set equation is solved explicitly, which requires very small time steps for stable computation and hence the entire iterations are rather time-consuming. An implicit approach [24] would be more efficient to reduce the number of iterations by relaxing the stability constraint, but it is difficult to be parallelized since it solves a matrix inverse problem. Other methods such as narrow band [25] and Multigrid [26] could improve the computation speed by reducing the computing domain. However, in GPU implementation, these methods need to use complex virtual memory paging schemes for handling the irregular computing domain, resulting in low performance due to the CPU-GPU communication latency and GPU memory management latency [27].

We employ the LBM to solve the level set equation, i.e. Eq. (1), which can be considered as a diffusion equation with an external force term. Since Eq. (1) does not have the nonlinear advection term compared with the NS equations, the lattice nodes can be simply connected by only the orthogonal links leading to a D3Q7 model (Fig. 3). Let $h_i(\boldsymbol{x},t)$ be the distance distribution function along the direction of $\boldsymbol{e}_i(\boldsymbol{x},t)(i=0,1,\ldots,6)$, the lattice Boltzmann equation for $h_i(\boldsymbol{x},t)$ reads

$$h_i(\boldsymbol{x}+\boldsymbol{e}_i\Delta t,t+\Delta t)=h_i(\boldsymbol{x},t)-[h_i(\boldsymbol{x},t)-h_i^{eq}(\boldsymbol{x},t)]/\tau_\phi+\Delta t\cdot F_i(\boldsymbol{x},t) \tag{2}$$

where $h_i^{eq}(\boldsymbol{x},t)=\phi(\boldsymbol{x},t)/7$, $F_i(\boldsymbol{x},t)=\beta g/7$ and $\tau_\phi=0.5+3g\Delta t$. After each step of collision and streaming, the distance function is obtained by $\phi(\boldsymbol{x})=\sum_i h_i(\boldsymbol{x})$ and the corresponding distance distribution function and equilibria, $h_i(\boldsymbol{x},t)$ and $h_i^{eq}(\boldsymbol{x},t)$, are updated by $h_i(\boldsymbol{x},t)=h_i^{eq}(\boldsymbol{x},t)=\phi(\boldsymbol{x},t)/7$. Boundary conditions are simply handled by the bounce-back at image edges. Through the Chapman-Enskog analysis [28], Eq. (2) recovers Eq. (1). At artery boundary, there exists a large gradient of the grayscale values so that $g\to 0$ and $\phi(\boldsymbol{x})$ converges on the boundary. Since g and Δt are both greater than zero, $\tau_\phi>0.5$ is guaranteed, which is required for the stability of LBM simulation [29]. The LBM has no strong stability restriction on Δt, large iteration steps are allowed to achieve fast computation in contrast to the explicit finite methods.

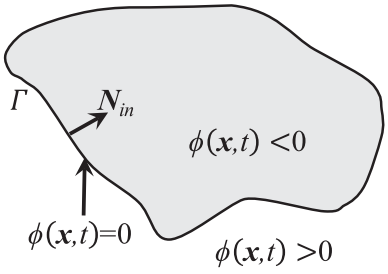

Fig. 2. Signed distance field.

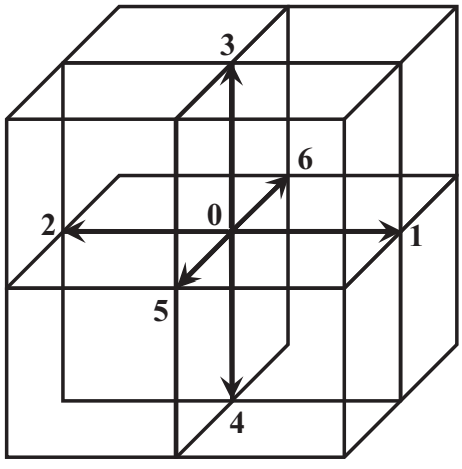

Fig. 3. D3Q7 lattice model.

2.1.2. Volumetric lattice Boltzmann method (VLBM) for fluid dynamics

The volumetric LBM (VLBM) was specifically developed for solving fluid dynamics in arbitrarily moving boundaries [30]. In this work, we consider rigid pore structure, thus the formation is simplified than that in the original method. In VLBM, the fluid particles are uniformly distributed in lattice cells, as opposed to sitting at lattice nodes in conventional node-based LBM. We introduce $n_i(\boldsymbol{x},t)$ representing particle population in cell $\boldsymbol{x}$ at time t with velocity $\boldsymbol{e}_i$ and deal with the time evolution of particle population,

$$n_i(\boldsymbol{x}+\boldsymbol{e}_i t, t+\Delta t) = n_i(\boldsymbol{x},t) - [n_i(\boldsymbol{x},t) - n_i^{eq}(\boldsymbol{x},t)]/\tau \tag{3}$$

where $n_i^{eq}(\boldsymbol{x},t)$ is the equilibrium particle population, $i(=0,1,\ldots,b)$ is the predefined directions of molecular motion, and τ is relaxation time. The $n_i^{eq}(\boldsymbol{x},t)$, $\boldsymbol{e}_i$ and b. pend on the selection of lattice model. In this work, we use the popular 3-D lattice with 19 molecular directions (b = 18), called D3Q19 model (Fig. 4). The discrete molecular velocities are

$$\boldsymbol{e}_i = \begin{cases} (0,0,0)c, & i=0 \\ (\pm1,0,0)c,(0,\pm1,0)c,(0,0,\pm1)c, & i=1-6 \\ (\pm1,\pm1,0)c,(\pm1,0,\pm1)c,(0,\pm1,\pm1)c, & i=7-18 \end{cases} \tag{4}$$

The equilibria of particle population are

$$n_i^{eq}(\boldsymbol{x},t) = N\omega_i\left(1+\frac{3\boldsymbol{e}_i\cdot\boldsymbol{u}}{c^2}+\frac{9(\boldsymbol{e}_i\cdot\boldsymbol{u})^2}{2c^4}-\frac{3\boldsymbol{u}^2}{2c^2}\right) \tag{5}$$

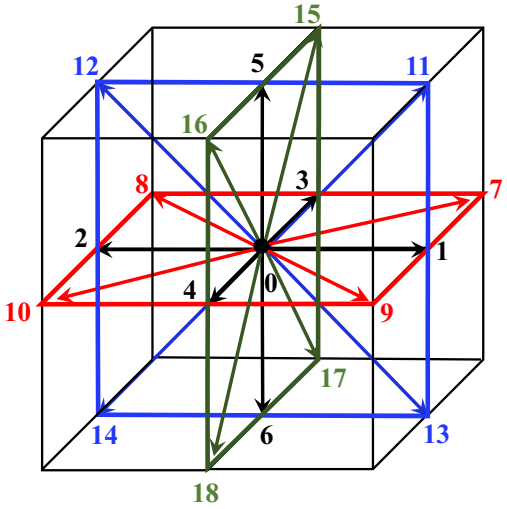

Fig. 4. D3Q19 lattice model.

where $c = \Delta\boldsymbol{x}/\Delta t = 1$ in lattice units. The weighting factors ω_i are $\omega_0 = 1/3$, $\omega_{1-6} = 1/18$, and $\omega_{7-18} = 1/36$. The mass and momentum conservations are strictly enforced by $N(\boldsymbol{x},t) = \sum_{i=0}^{b} n_i(\boldsymbol{x},t)$ and $\boldsymbol{u}(\boldsymbol{x},t) = \sum_{i=0}^{b}\boldsymbol{e}_i n_i(\boldsymbol{x},t)/N(\boldsymbol{x},t)$, respectively.

Similar to the concept of volume of fluid (VOF) [31] used to track interfaces between two fluids, the VLBM uses the occupation of solid volume $\mathrm{P}(\boldsymbol{x})$, defined by the ratio of solid volume $\Delta V_S(\boldsymbol{x},t)$ to lattice volume $\Delta V(\boldsymbol{x},t)$, to express the arbitrary boundary. As shown in Fig. 5(a), the solid black curve depicts a boundary separating fluid (empty) and solid (shaded) domains. Cells adjacent to the boundary may be occupied entirely by either solid or fluid, while others may have partial fluid and partial solid volumes. In the whole domain, cells are categorized through three distinct cell types: fluid cell ($\mathrm{P}=0$), solid cell ($\mathrm{P}=1$), and boundary cell ($0<\mathrm{P}<1$). It is noted that the relation between the particle distribution function f_i in the conventional node-based LBM and the particle population n_i in the VLBM is $f_i(\boldsymbol{x},t) = n_i(\boldsymbol{x},t)/[1-\mathrm{P}(\boldsymbol{x})]$ taking the total volume of a cell to be unity. Unlike VOF that constructs a convection equation for the volume ratio function to track the interface, the VLBM divides the time evolution equation, Eq. (3), into two processes taking into account the arbitrary boundary through $\mathrm{P}(\boldsymbol{x})$.

- Collision:

$$n_i'(\boldsymbol{x},t) = n_i(\boldsymbol{x},t) - [n_i(\boldsymbol{x},t) - n_i^{eq}(\boldsymbol{x},t)]/\tau \tag{6}$$

where $n_i'(\boldsymbol{x},t)$ represents the "post-collision" particle population. For a rigid boundary, this process is similar to node-based LBM.

- Streaming: In boundary cells, only an appropriate volume fraction of fluid particles will be able to stream to its neighboring cell. A streaming in VLBM consists of two operations: (i) streaming from its upwind neighboring cells, $[1-\mathrm{P}(\boldsymbol{x})]\cdot n_i'(\boldsymbol{x}-\boldsymbol{e}_i\Delta t,t)$ and (ii) bounce-back from the downwind cells, $\mathrm{P}(\boldsymbol{x}+e_{i^*}\Delta t)\cdot n_{i^*}'(\boldsymbol{x},t)$ where i^* corresponds to the direction opposite to the i^{th} direction $\boldsymbol{e}_{i^*} = -\boldsymbol{e}_i$. Thus, particles in cell $\boldsymbol{x}$ at time $t+\Delta t$ after a streaming operation are as follows

$$n_i''(\boldsymbol{x},t+\Delta t) = [1-\mathrm{P}(\boldsymbol{x})]n_i'(\boldsymbol{x}-\boldsymbol{e}_i\Delta t) + \mathrm{P}(\boldsymbol{x}+\boldsymbol{e}_{i^*}\Delta t)n_{i^*}'(\boldsymbol{x},t) \tag{7}$$

This modified streaming process ensures that particles are advected to their appropriate places in the fluid domain but doesn't introduce any extra mass. In fluid cells where $\mathrm{P}(\boldsymbol{x})=0$, Eq. (7) reduces to the counterpart of node-based LBE where only upwind streaming occurs.

2.1.3. Seamless connection between image segmentation and CFD

It is noted that in VLBM all the equations together with no-slip boundary condition are self-regularized through the solid occupa-

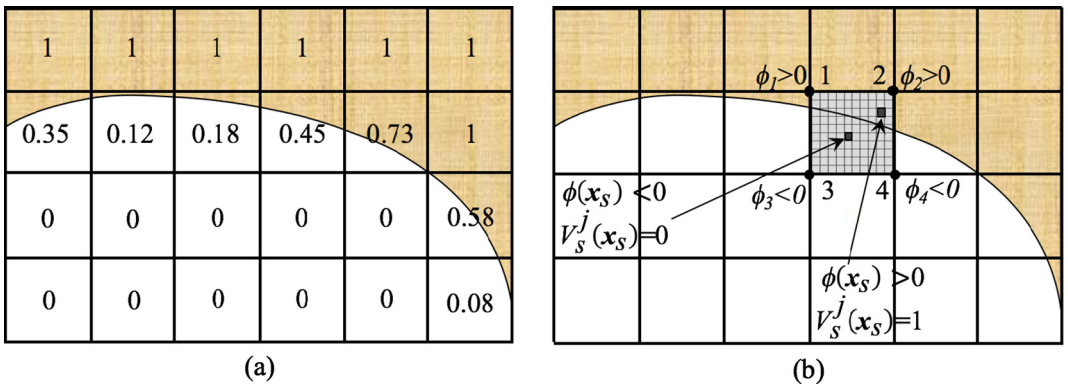

Fig. 5. (a) Illustration of three types of cells in VLBM represented by solid volume ratio P: $\mathrm{P}=0$ (fluid cell), $\mathrm{P}=1$ (solid cell), $0<\mathrm{P}<1$ (boundary cell). (b) Schematic of P-value calculation for a boundary cell using signed distance field ϕ obtained from the image sigmentation.

tion function $P(\boldsymbol{x})$. Thus, calculating $P(\boldsymbol{x})$ of each cell in the entire domain after the image segmentation is the key to the success of seamless connection between image segmentation and CFD. It turns out that the calculation of $P(\boldsymbol{x})$ from the signed distance field $\phi(\boldsymbol{x})$ is rather easy. In what follows, we use 2-D mesh as an example to illustrate the algorithm to calculate $P(\boldsymbol{x})$. In the distance field $\phi(\boldsymbol{x})$, the zero level set of $\phi(\boldsymbol{x}) = 0$ represents the wall of the pores. The sign of ϕ at each lattice node indicates whether the node is in solid or fluid region. For a particular cell with four nodes, if all four ϕ values are negative, the cell is a fluid cell with $P = 0$. If all four ϕ values are positive, the cell is a solid cell with $P = 1$. The remaining cells are boundary cells with mixed positive and negative ϕ values at the four nodes. This is because a boundary cell is occupied by partial solid and partial fluid.

Take an example of the grey cell in Fig. 5(b), we now described the two steps about how to calculate the solid volume ratio $P(\boldsymbol{x})$ in the boundary cell located at $\boldsymbol{x}$, of which ϕ_i at each node i (i = 1–4), either positive or negative, is known after the image segmentation. First, uniformly divide the cell into a refined grid with q^2 sub-cells. The $\phi(\boldsymbol{x}_s)$ value of each sub-cell can be interpolated from the ϕ values at the four nodes based on its center location $\boldsymbol{x}_s$. Second, we calculate the total solid volume in the boundary cell. If the j^{th} sub-cell is solid with $\phi(\boldsymbol{x}_s) > 0$, we set $V_s^j(\boldsymbol{x}_s) = 1$. Otherwise, we set $V_s^j(\boldsymbol{x}_s) = 0$. The solid volume ratio of the boundary cell is calculated as $P(\boldsymbol{x}) = \sum_{j=1}^{q^2} V_s^j(\boldsymbol{x}_s)/q^2$. One should be noticed that the grid refinement of q^2 inside a boundary cell determines the accuracy of $P(\boldsymbol{x})$. A larger q results in more accurate $P(\boldsymbol{x})$ value but higher computational cost. It is noted that this grid refinement is only used for calculating the solid volume ratio in boundary cells, $P(\boldsymbol{x})$.

Once $P(\boldsymbol{x})$ is calculated from the segmentation, CFD can be kicked in seamlessly. Fig. 6 shows the flow chart of seamless connection between image segmentation and CFD.

2.2. GPU parallel computing

It has been well demonstrated that LBM is ideally suited for GPU parallel computing. Recently, our efforts have been on GPU-accelerated LBM for turbulent and biomedical flows [32,33]. Since the GPU parallelism for image segmentation and fluid dynamics is similar, here we express the algorithm using VLBM as an example. In this work, we only implement parallelism on single NVIDIA GPU card. Specifically, C++ is used for the CPU part and CUDA for the GPU part.

As shown in Eqs. (3), (6) and (7), in VLBM, the time evolution of particle population is divided into two operations — collision and streaming, which are self-regularized through $P(\boldsymbol{x})$. The bounce-back boundary condition at porous structure is taken into account in the streaming process, i.e. Eq. (7). Thus, the challenging part to deal with complicated geometry and its corresponding boundary condition on structure are avoided. Only boundary conditions at inlet and outlet need to be treated. Furthermore, we can perform the same parallelization for both boundary and fluid cells, $P(\boldsymbol{x}) < 1$, to minimize program branches thus to improve the parallel efficiency.

The tasks for collision and streaming are independent thus executed successively. For convenience, we introduce $n_i'(\boldsymbol{x}, t)$ as intermediate variable after collision, called "post-collision", as shown in Eq. (6). In *collision*, computing $n_i'(\boldsymbol{x}, t)$ only requires local information within the cell thus the execution is carried out in one scalar processor. In *streaming*, Eq. (6) involves two parts, one is streamed from upwind neighboring cells and another bounce-back from the downwind neighboring cells, as indicated in Eq. (7). We use "pull back" scheme in which write is aligned and read is misaligned for high efficiency [34]. Fig. 7 shows a schematic of pull back scheme on the blue plane in D3Q19 model (Fig. 4) with upwind streaming in black and bounce-back in red.

The flow chart of GPU parallelism for VLBM is shown in Fig. 8(a). The shaded lines are implemented in the GPU device. The iteration loop, schematized in Fig. 8(b), is executed in one kernel. To minimize the data transfer between CPU host and GPU device and maximize the computational efficiency, we implement the major computation on GPU device including declaration and initiation of flatten forms and iteration of VLBM and merely left input and output on CPU host. The GPU algorithm is based on our previous work [32]. We make efforts to improve the computation efficiency through optimization dynamic allocation, coalesced global memory, register arrangement, and pragma unroll.

2.2.1. Dynamic allocation

Memory arrangement in both GPU device and CPU host is of importance to overcome the limits of CPU static zone memory

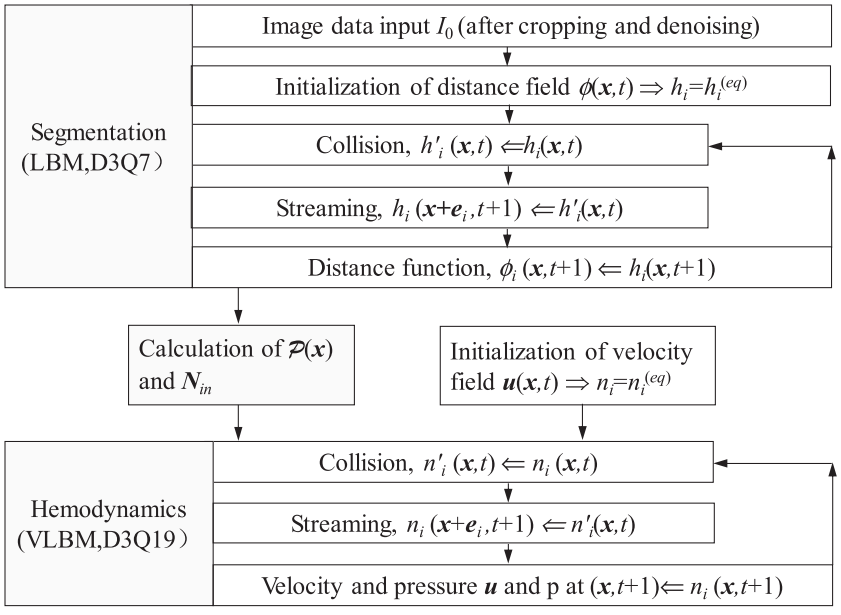

Fig. 6. Computational flow chart demonstrate a seamless connection between images segmentation and CFD.

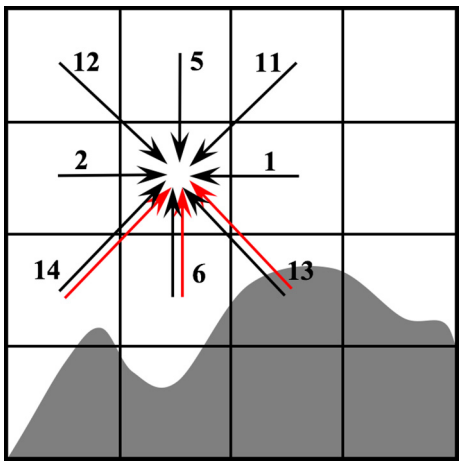

Fig. 7. Illustration of "pull back" scheme on the blue plane in D3Q19 model (Fig. 4) with upwind streaming in black and bounce-back in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

and GPU device memory. In GPU device, we employ the ordering method called Structure of Array format [32]. Nowadays, GPU cards are capable of storing anywhere from 3 to 6 GB of data in memory. In order to fully utilize this space, the CPU part of this algorithm relies on dynamic (heap) memory as opposed to statically allocated (stack) memory. There are typically file and static memory limits imposed by either an operating system or system administrator that cap at about 2 GB. If an uncompiled code were to attempt to use static allocation to utilize the full capacity of a GPU, it would run into the 2 GB size limit as the executable created by the compiler would either need to be larger than the file size limit or exceed the static memory limit. Dynamic allocation allows the operating system to provide as much memory as what is available to provide by the CPU RAM. The memory available in CPU RAM is typically many times the maximum memory capacity of GPU RAM, so algorithm can fully utilize the memory capacity of a GPU card if a dynamic memory is declared. For the purposes of data copy between CPU host and GPU device, these arrays should contain memory that is logically contiguous. C++ provides a means to allocate logically contiguous dynamic memory using the 'new' keyword. The 'new' keyword is limited, however, to dynamically allocating one dimensional arrays. We create a class in C++, which would dynamically allocate a contiguous block of memory for regular, multidimensional arrays using the 'new' keyword. The class would then internally translate the user's request for an element in 2–4 dimensions to the equivalent request in 1 dimension. This class is supposed to easily pass its underlying data to the GPU and then easily retrieve the information from the GPU and to manage allocation and deallocation of memory in a safe way that required little to no effort from the user in order to ease the burden on the user.

2.2.2. Coalesced global memory

Global memory is an area in the off-chip device memory available for all the thread blocks, through which the GPU can communicate with the host CPU as well as for the data input to and output from kernels. The $n_i(\boldsymbol{x},t)$, $n_i'(\boldsymbol{x},t)$, geometry information $\mathrm{P}(\boldsymbol{x})$ and hydrodynamic parameters are stored in the global memory, which

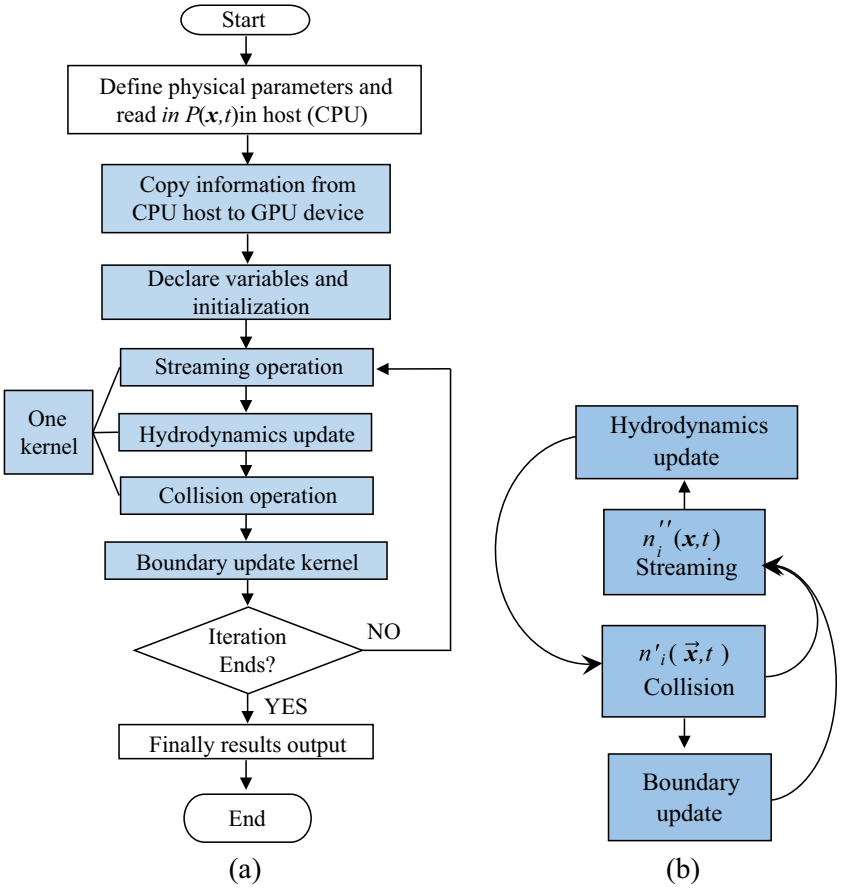

Fig. 8. Flow chart of GPU parallelism of VLBM. (a) VLBM work flow. (b) The iteration loop in GPU device.

should be used in every evolution step. The bandwidth of global memory, however, is a significant limit to accelerate calculation speed. To apply bandwidth effectively, we use 19 arrays to store the different data in the 19 directions of Fig. 4, just like the "1" direction, all n_1s in the domain storing in an array, as shown in Fig. 9. In the GPU, warp (32 threads) is unit to run together and half-warp is a unit to call for global memory. If the memory address is continued and the visiting memory of a warp is lower than the specific facility limit, the fast access can be achieved at one time.

2.2.3. Register arrangement

Even though the coalesced global memory access is designed in the code, it is still obviously lower than the on-chip memory. In most cases, accessing a register consumes zero clock cycles per instruction [35]. Therefore, we try to decrease the visiting time to global memory by combining streaming operation, hydrodynamic update and collision operation into a kernel. In this kernel, the needed parameters are defined in the limited register memory (usually 32-bit per register) and read in from global memory for the following calculations.

2.2.4. Pragma unroll

In the GPU, branch conflict is also a factor about calculation speed, resulting from special structure of streaming multiprocessor and strict parallelism in a half-warp. Unrolled loop can effectively decrease the branch conflict. As previous introduction, 19 arrays are defined to separate 19 directions, which is also used to unroll the loop. In addition, the calculation of equilibrium function in the collision part and boundary update is also unrolled to fit 19 directions.

2.3. Pore-structure upscaling

Upscaling is a typical technique used in reservoir simulators to reduce computation size for uncertainty analysis and risk assessment [36,37]. It assigns "effective" properties such as permeability on coarse grid from the properties on fine grid through arithmetic

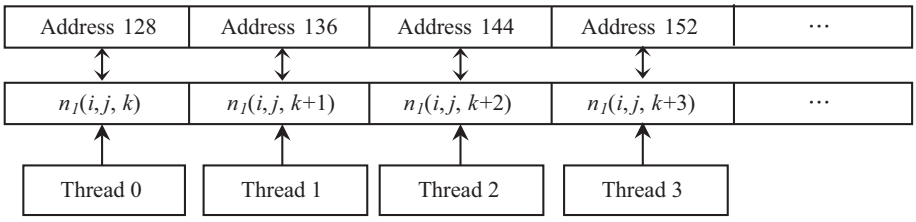

Fig. 9. Examples of coalesced global memory access pattern.

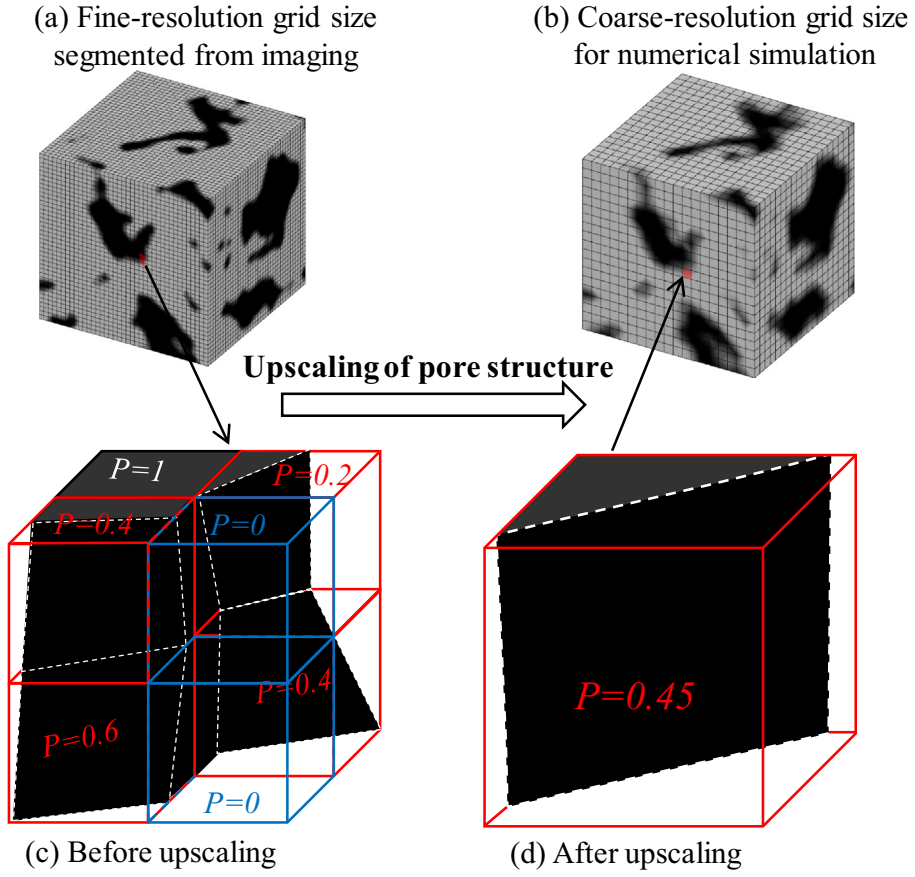

Fig. 10. Upscaling of pore structure from (a) high-resolution imaging data to (b) low-resolution mesh. The porosity of the pore structure remains exact the same in (c) before and (d) after upscaling.

average, harmonic average, geometric average and their combinations. As a result, one can perform analysis and assessment through the coarse grid to reduce the computation. We design this technique to substitute the fine resolution pore-structure segmented from imaging data with a coarse resolution pore structure through an average of $P(\boldsymbol{x})$. The pore-structure upscaling is featured with (1) volume preservation, (2) similarity between geological and simulation grid, and (3) preservation of geological and physical properties. The current radiological imaging resolution through micro-CT is around 4 μm. If we want to make comparable numerical study of PSPMF to laboratory experiment with typical sample size of 10 mm^3, the computation grid size will be above 2000^3, which exceeds the research capability in terms of the computation cost and memory requirement. The objective of pore-structure upscaling is to reduce the computation grid size while maintain equivalent pore structure and identical porosity.

Fig. 10 illustrates the pore-structure upscaling technique from fine–resolution (a) to coarse-resolution (b) for a target porous media. An example to upscale 8 cells to one cell through the average of P values is shown in (c) and (d). From the image segmentation, the fine pore structure (a) is represented by the solid ratio function $P(\boldsymbol{x})$. Each mesh cell is labeled by a P value distinguishing solid cell ($P = 1$, black, filled), fluid cell ($P = 0$, blue, empty), boundary cell ($0 < P < 1$, red, partial filled), seen in Fig. 10(c). To upscale 8 times, we group the fine resolution mesh into blocks, each of which contains 8 cells and each block is substituted by one single cell with the same volume and average P value of 8 cells. If the block contains all fluid or solid cells, the substitute cell will remain the fluid or solid cell. If the block includes boundary cell (s), the upscaling results in a big boundary cell. In Fig. 11, we show an example for two successive 8-times upscaling to (b) and (c) from fine-resolution pore-structure (a). The pore-structure upscaling guarantees identical porosity. After the primary upscaling from (a) to (b), although the grid becomes coarse, the feature of pore structure is well retained thus other macroscale parameters can be well reserved. However, further upscaling may cause a merge of pores resulting in a distortion of the pore structure; see the red area in Fig. 11(c) where two separate pores are connected. It should be noted that the degree of pore-structure upscaling is limited by resolution of imaging as well as the shortest distance of separate pores.

The significance of the pore-structure upscaling is that one can do low computation cost simulation while retain the features of pore structure from high-resolution imaging data. Plus, the GPU acceleration, becomes one of the most powerful computational tools for PSPMFs. We use the following example to demonstrate the power of this simulation method. A cylindrical core plug drilled from a quartz-made stone has a dimension of 4 mm length and 1 mm diameter. Fig. 12 shows its image scanned from a micro-CT with gray values from 0 to 255. If the typical computation capacity

Fig. 12. Gray image of a cylindrical core imaged by micro-CT.

is 256^3 on a local workstation, one can easily upgrade to 512^3 through GPU parallelism. With the pore-structure upscaling, one becomes capable to deal with 1024^3 assuming no significant pore mergence occurs. That is to say, with the combination of GPU and pore-structure upscaling, with the same computation cost, one can solve PSPMF in 64 times finer pores. From another point view, if the scanning resolution remains the same, one can study PSPMF in a cylindrical core with 64 times larger dimension. Each of them results in significant advancement to unveil the micro to macro behavior of rock-fluid systems.

3. Application studies

In order to demonstrate the reliability and applicability, we now conduct two application studies. One is a 3-D pipe flow driven by a constant pressure gradient. Another is a PSPMF in a digital core. The computation tasks are carried out on a NVIDIA Tesla K20 GPU card, which has 2496 CUDA cores with 706 MHZ clock frequency and 5 GB of global memory. To evaluate the GPU parallel performance, the serial computation is performed with Intel x86_64 CPU, which has 16 computing cores with 1.87 GHZ and 64 GB DDR3 Random Access Memory (RAM).

3.1. Three-dimensional pipe flow

We consider an oil flow in a pipe with length L = 0.2 mm and diameter D = 0.1 mm. The density and viscosity are ρ = 900 kg/

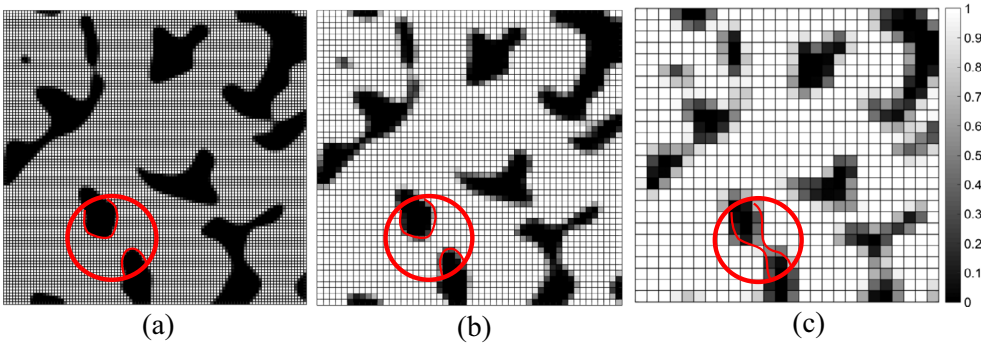

Fig. 11. An example of pore-structure upscaling. (a) Fine-resolution pore-structure. (b) 8-times upscaling. (c) 64-times upscaling.

m^3 and μ = 0.0135 Pa s, respectively. The flow is along z direction. Since oil flow in porous media is usually under low Reynolds number, we use a constant pressure gradient dp/dz = 2160 Pa/m to drive the flow, of which the Reynolds number (Re) is 6.67e-4. Such a flow has an analytical solution as

$$u_z = -\frac{1}{4\mu}\frac{dp}{dz}(R^2 - r^2) \quad (0 \leqslant r \leqslant R) \tag{8}$$

We apply for periodic boundary condition at inlet and outlet and no-slip boundary condition at wall.

Since boundary of a pipe can be mathematically formulated, the distance field $\phi(\boldsymbol{x}, t)$ can be directly calculated. The solid volume ratio function $\mathrm{P}(\boldsymbol{x})$ for all the cells can be determined relatively easy (no need to solve the level set equation). Following are steps to calculate $\mathrm{P}(\boldsymbol{x})$. For the sake of simplicity, we use one layer of the mesh (2-D) to illustrate, as seen in Fig. 13.

(1) **Calculate distance of each node** (x, y, z) to the center O (0, 0, 0), i.e.$\phi = \pm\sqrt{x^2 + y^2 + z^2}$ Select negative sign for $\phi < R$ and positive sign for $\phi > R$. If $\phi = R$, assign $\phi = 0$.

(2) **Distinguish cell type through the signs of ϕ at all four nodes** If all ϕ values are negative, the cell is fluid with $\mathrm{P} = 0$. If all four ϕ values are positive, the cell is solid and $\mathrm{P} = 0$. Otherwise, the cell is a boundary cell occupied by partial solid and partial fluid.

(3) **Calculate P values for boundary cells (shaded) using the method expressed in Section 2.1.3** The algorithm for determination of $\mathrm{P}(\boldsymbol{x})$ can be validated by the volume of the pipe. Since P value is defined as the solid ratio in a grid cell (a cube in 3-D), the total volume of the fluid channel can be obtained by $V_p = \sum_{\boldsymbol{x}}[1 - \mathrm{P}(\boldsymbol{x})]$ whereas the analytical volume of the pipe can be calculated by $V_a = \pi(D/2)^2 L$. Table 1 shows the relative error of pipe volume at four resolutions. It is seen that the distance field method is highly accurate even with a relatively low resolution. In VLBM, $\mathrm{P}(\boldsymbol{x})$ is directly fed in as an input together with other initial computation set-up followed by iteration of collision and streaming till the flow reaches a steady state. Fig. 14 shows the comparison of velocity profile along diameter with resolution $50 \times 50 \times 100$. It can be seen that the VLBM simulation (symbols) captures nearly identical velocity profile to the analytical solution (solid line).

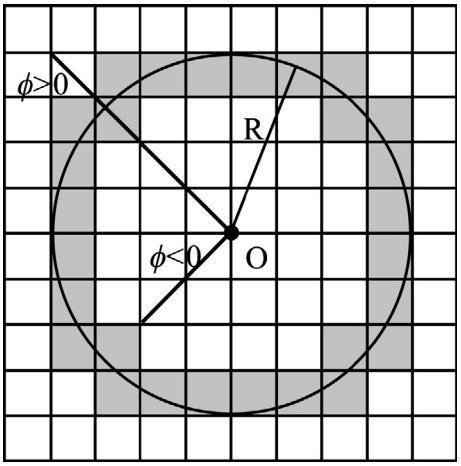

Fig. 13. Determination of P-value for each cell based on the signed distance function ϕ at the nodes of the cell: $\mathrm{P} = 1$ if all ϕs are positive, solid cell (outside the circle); $\mathrm{P} = 0$ if all ϕs are negative, fluid cells (inside the circle); and $0 < \mathrm{P} < 1$ if patial positive and patial negative of ϕs, boundary cell (shaded).

Table 1
Relative error P value calculation using level set method. Vp and Va are volumes of the pipe calculated by P values and analytical formula.

Grid Size	$20^2 \times 40$	$40^2 \times 80$	$50^2 \times 100$	$100^2 \times 200$
(Vp-Va)/Va (%)	0.26	0.084	0.045	0.015

3.2. Pore scale flow in a digital core

In this part, we indicate the efficiency of GPU parallelism and the applicability of pore-structure upscaling in the course of solving pore-scale flow in a sample digital core. A cylindrical core (Fig. 11) is first scanned through micro-CT (MicroXCT-400) with a resolution 0.37 μm/pixel. Since different phases have different X-ray absorption, the output of the scanning is a gray image. A 256^3 (pixel^3) cube is cut from the original image for the cylindrical core, as shown in Fig. 15. Gauss smoothing is first applied for reducing the effect of salt-and-pepper noise in the raw image. Then the image is segmented using LBM described in Section 2.1.1. After the segmentation, the $\mathrm{P}(\boldsymbol{x})$ field is calculated using the algorithm illustrated in Section 2.1.3 with an example in 3.1. As shown in Fig. 6, the $\mathrm{P}(\boldsymbol{x})$ field together with initial/boundary conditions and physical properties are fed in VLBM to solve fluid dynamics.

We select a representative plane from the digital core (Fig. 15) to show the outcome of image segmentation. In Fig. 16, we compare the segmented pore structure (b) to the scanned image (a).

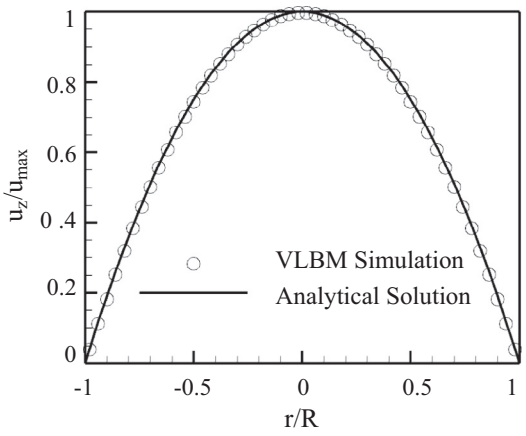

Fig. 14. Comparison of velocity profile between VLBM simulation and analytical solution.

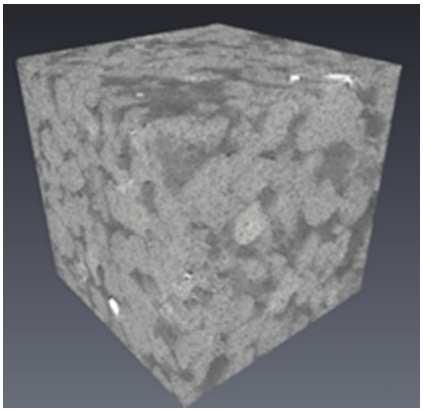

Fig. 15. A 256^3 (pixel^3) cube cut from micro-CT scanning data.

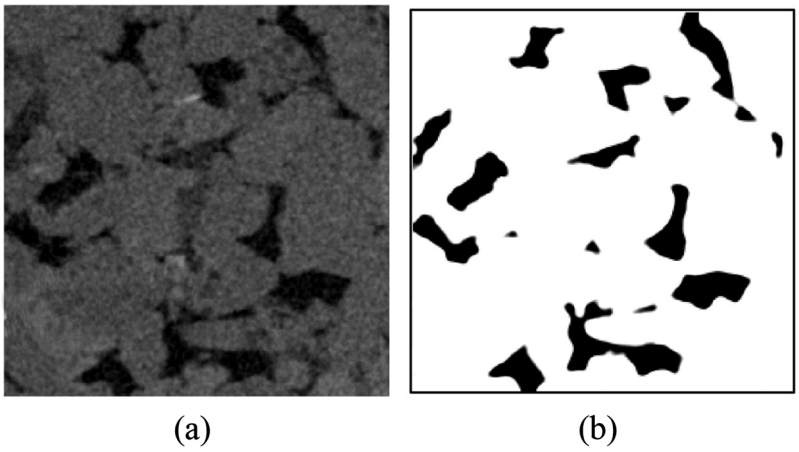

Fig. 16. Scanned (a) vs. segmented (b) pore structure.

It is clearly seen that the result has well captured the pores (in black).

To validate the pore-structure upscaling, we upscale the resolution from 256^3 to 128^3, 85^3, 64^3, and 32^3 and examine the pressure distribution and permeability on each upscale. Fig. 17 shows the normalized plane-averaged pressure along the flow stream. The downstream pressure distributions calculated from the original grid size and four upscaled grid sizes collapse, indicating that the pore-structure upscaling can quite well retain the driving mechanism for the PSPMF. Permeability is calculated by using Darcy's equation. In Table 2, it is seen that the primary pore structure upscaling to 128^3 (8 times) remains the permeability identical. The secondary upscaling to 64^3 (64 times) deviates the permeability about 28% from the original 256^3, which is acceptable considering inaccurate boundary conditions [38]. The 128 times upscaling to 32^3 results in 227% larger permeability, implying that pore-structure upscaling is not unlimited.

The GPU acceleration is examined in Table 3 for three resolutions though MLUPS (Million Lattice node Updates Per Second) to represent the performance of GPU parallelism. For the model with lattice size 256^3, our GPU code achieves MLUPS as high as 450.7, accelerating 965.1 times compared with the CPU serial computation.

The MLUPS is lower than classic lid-driven cavity flow model [39] (no complicated geometry involved) but faster than porous media models with same hardware condition [40]. Considering

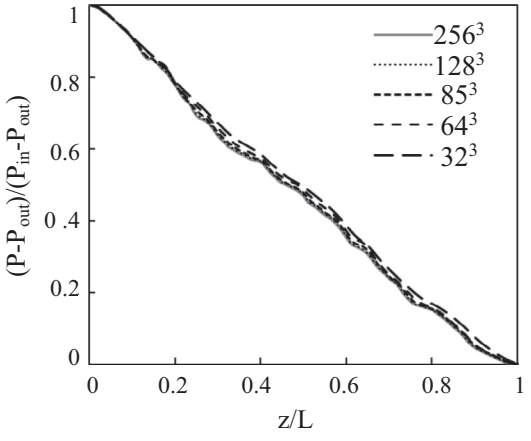

Fig. 17. Normalized plane-averaged pressure along downstream at original and 4 upscales.

Table 2
Permeability at original and 3 upscaless.

Grid Size	256^3	128^3	85^3	64^3	32^3
Permeability (mD)	1.53	1.53	1.65	1.95	5.01

Table 3
GPU performance on three grid resolutions.

Grid Size	GPU-parallelism		CPU-serial		Speed up of GPU-parallelism vs. CPU-serial
	Steps/min	MLUPS	Steps/min	MLUPS	
64^3	96519	421.7	304.4	1.33	317.1
128^3	12689	443.5	31.8	1.11	399.5
256^3	1612	450.7	1.67	0.467	965.1

that a single 5 GB GPU card, of which many computation labs can possess, such a speed-up means a significant advance in the capability of computational pore-scale fluid dynamics.

4. Summary and discussion

We have presented a novel and powerful computational method for studying image-based PSPMFs. A unique integration of mesoscopic modeling with GPU parallelism and pore structure upscaling is developed. The unified LBM modeling of image segmentation and fluid dynamics enables seamless connection between image processing and CFD through the solid volumetric function, $P(\boldsymbol{x})$, which can be calculated from the signed distance field after the image segmentation. This feature not only eliminates the ad-hoc coalition of software packages for geometry reconstruction and mesh generation to connect the image segmentation and CFD tasks, but also enables unified GPU implementation to achieve significant speed-up of computation. This method represents an important advance in studying PSPMFs by avoiding laborious, complex, and error-prone processes in modeling and computation and will benefit the entire community of porous media flow. Two application studies are conducted. The first, 3-D pipe flow, demonstrates the reliability of this system for image representation. The pipe volume calculated from the signed distance field is in good agreement with its analytical volume, 5 μm/pixel and 1 μm/pixel resulting in relative errors of 0.26% and 0.015% respectively as seen in Table 1. The velocity profile simulated by VLBM is nearly identical to the analytical solution. The second, pore scale flow in a digital core, demonstrates the applicability to PSPMF in subsurface hydrology engineering and its computational efficiency. Based on

the accuracy segmentation result for the complicated sample, the pore-structure upscaling reduces 8–16 times of computation cost, i.e. from 256^3 to 128^3–64^3, but the simulation results of pressure distribution and permeability remain the same. Third, GPU parallelism achieves nearly 450 MLUPS on a single 5 GB GPU card, which significantly expand the computation capability for real application in the field of porous media flow.

It is noted that the systematic method introduced above is sustainable to introduce more physical models such as interfacial dynamics for multiphase flows, fluid–structure interaction for deformable pore structure, non-Newtonian effects, etc. to meet the need of real-world application. However, more sophisticated modeling means higher computational expense. We are currently working on the multi-GPU-card implementation. We will focus on the critical steps for implementing asynchronous data communication between GPUs, which includes transfer between the compute nodes. Based on this operation, 1024^3 size model can be calculated by using 64 nodes, and the calculation speed can be improved to more than 2000 MLUPS.

Acknowledgements

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is support by (1) International Research Development Fund (IRDF) of IUPUI, (2) the National Natural Science Foundation of China (Nos. 51490654, 51711530131, 51674280), and (3) Program of Innovation Projects (YCXJ2016021).

Conflict of interest

The authors declared that there is no conflict of interest.

References

[1] S. Whitaker, Flow in porous media I: A theoretical derivation of Darcy's law, Transp. Porous Media 1 (1986) 3–25.
[2] Q. Kang, P.C. Lichtner, D.R. Janecky, Lattice Boltzman method for reacting flows in porous media, Adv. Appl. Math. Mech. 2 (2010) 545–563.
[3] G. Lu, D.J. DePaolo, Q. Kang, D. Zhang, Lattice Boltzmann simulation of snow crystal growth in clouds, J. Geophys. Res. Atmo. (2009) D07305.
[4] R.A. Ketcham, Computational methods for quantitative analysis of three-dimensional features in geological specimens, Geosphere 1 (2005) 32–41.
[5] O. VanGeet, Best Practices Guide for Energy-efficient Data Center Design, EERE Publication and Product Library, 2010.
[6] M.A. Knackstedt, A.P. Sheppard, M. Sahimi, Pore network modelling of two-phase flow in porous rock: the effect of correlated heterogeneity, Adv. Water Res. 24 (2001) 257–277.
[7] B. Costa, P. Courat, Molecular dynamic study of fluid flow through a porous medium, Braz. J. Phys. 25 (1995) 252–256.
[8] M.B. Liu, G.R. Liu, Smoothed particle hydrodynamics (SPH): an overview and recent developments, Arch. Comput. Methods Eng. 17 (1) (2010) 25–76.
[9] M. Liu, P. Meakin, H. Huang, Dissipative particle dynamics simulation of pore-scale multiphase fluid flow, Water Resour. Res. 43 (4) (2007) W04411.
[10] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, Ann. Rev. Fluid Mech. 30 (1) (1998) 329–364.
[11] C.K. Aidun, J.R. Clausen, Lattice-Boltzmann method for complex flows, Ann. Rev. Fluid Mech. 42 (2010) 439–472.
[12] Y. Huidan et al., Mass-conserved volumetric lattice Boltzmann method for complex flows with willfully moving boundaries, Phys. Rev. E: Stat., Nonlin, Soft Matter Phys. 89 (6) (2014), 063304-063304.
[13] H. Liang, P. Cheng, Lattice Boltzmann simulations of anisotropic permeabilities in carbon paper gas diffusion layers, J. Power Sources 186 (1) (2009) 104–114.
[14] C. Obrecht, F. Kuznik, B. Tourancheau, J.-J. Roux, Scalable lattice Boltzmann solvers for CUDA GPU clusters, Parallel Comput. 39 (6) (2013) 259–270.
[15] X. He, L. Luo, Theory of lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation, Phys. Rev. E 56 (12) (1997) 6811–6817.
[16] M.A. Gallivan, D.R. Noble, J.G. Georgiadis, R.O. Buckius, An evaluation of the bounce-back boundary condition for lattice Boltzmann simulations, Int. J. Numer. Meth. Fluids 25 (3) (1997) 249–263.
[17] H.M. Kang et al., Variance component model to account for sample structure in genome-wide association studies, Nat. Genet. 42 (4) (2010) 348–354.
[18] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech. 30 (1998) 329–364.
[19] P. Bailey, J. Myre, S.D. Walsh, D.J. Lilja, M.O. Saar, Accelerating lattice Boltzmann fluid flow simulations using graphics processors, in: 2009 International Conference on Parallel Processing, IEEE, 2009, pp. 550–557.
[20] H. Liang, P. Cheng, Pore-scale simulations on relative permeabilities of porous media by lattice Boltzmann method, Int. J. Heat Mass Transf. 53 (9–10) (2010) 1908–1913.
[21] Z. Wang, Z. Yan, G. Chen, Lattice Boltzmann method of active contour for image segmentation, in: Image and Graphics (ICIG), 2011 Sixth International Conference on, IEEE, 2011, pp. 338–343.
[22] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.
[23] P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intell. 12 (7) (1990) 629–639.
[24] J. Weickert, B.M.T.H. Romeny, M.A. Viergever, Efficient and reliable schemes for nonlinear diffusion filtering, IEEE Trans. Image Process. 7 (3) (1998) 398–410.
[25] D. Peng, B. Merrimann, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[26] G. Papandreou, P. Maragos, Multigrid geometric active contour models, IEEE Trans. Image Process. 16 (1) (2007) 229–240.
[27] A. Lefohn, R. Whitaker, A GPU-based, three-dimensional level set solver with curvature flow, University of Utah technical report UUCS-02-017, 2002.
[28] S. Chapman, T.G. Cowling, The Mathematical Theory of Non-uniform Gases, Cambridge University Press, 1970.
[29] D. Wolf-Gladrow, Lattice-gas Cellular Automata and Lattice Boltzmann Models: An Introduction, Springer, 2000, p. 308.
[30] H. Yu et al., Mass-conserved volumetric lattice Boltzmann method for complex flows with willfully moving boundaries, Phys. Rev. E 89 (6) (2014) 063304.
[31] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[32] Z. Wang, N. Chen, Y. Zhao, H. Yu, GPU-accelerated Lattice Boltzmann method for extracting real biomechanical geometry and volumetric boundary condition, Comput. Fluids 115 (2015) 192–200.
[33] H.W. Yu et al., GPU accelerated lattice Boltzmann simulation for rotational turbulence, Comput. Math. Appl. 67 (2) (2014) 445–451.
[34] N. Delbosc, J.L. Summers, A. Khan, N. Kapur, C.J. Noakes, Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation, Comput. Math. Appl. 67 (2) (2014) 462–475.
[35] J. Reese, S. Zaranek, Gpu programming in matlab, in: MathWorks News&Notes, The MathWorks Inc, Natick, MA, 2012, pp. 22–25.
[36] M.A. Christie, M.J. Blunt, Tenth SPE comparative solution project: A comparison of upscaling techniques, Spe Reservoir Eval. Eng. 4 (4) (2001) 308–317.
[37] Y. Yang, C. Wang, J. Yao, Y. Gao, A new voxel upscaling method based on digital rock, Int. J. Multiscale Comput. Eng. 13 (4) (2015).
[38] C. Manwart, U. Aaltosalmi, A. Koponen, R. Hilfer, J. Timonen, Lattice-Boltzmann and finite-difference simulations for the permeability for three-dimensional porous media, Phys. Rev. E 66 (1) (2002) 016702.
[39] T. Tomczak, R.G. Szafran, Memory layout in GPU implementation of lattice Boltzmann method for sparse 3D geometries, arXiv preprint arXiv:1611.02445, 2016.
[40] G.U.O.Z. Zhu Lianhua, GPU accelerated lattice Boltzmann simulation of flow in porous media, Chinese J. Comput. Phys. 32 (1) (2015) 20–26.