Computer Physics Communications 217 (2017) 171–179

Contents lists available at ScienceDirect

Computer Physics Communications

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

GPU-accelerated red blood cells simulations with transport dissipative particle dynamics☆

Ansel L. Blumers [a], Yu-Hang Tang [b], Zhen Li [b,*], Xuejin Li [b], George E. Karniadakis [b,*]

[a] Department of Physics, Brown University, Providence, RI, USA
[b] Division of Applied Mathematics, Brown University, Providence, RI, USA

ARTICLE INFO

Article history:
Received 18 November 2016
Received in revised form 7 February 2017
Accepted 29 March 2017
Available online 18 April 2017

Keywords:
Dissipative particle dynamics
Red blood cell
GPU
Advection–diffusion–reaction
Mesoscopic modeling
Blood flow

ABSTRACT

Mesoscopic numerical simulations provide a unique approach for the quantification of the chemical influences on red blood cell functionalities. The transport Dissipative Particle Dynamics (tDPD) method can lead to such effective multiscale simulations due to its ability to simultaneously capture mesoscopic advection, diffusion, and reaction. In this paper, we present a GPU-accelerated red blood cell simulation package based on a tDPD adaptation of our red blood cell model, which can correctly recover the cell membrane viscosity, elasticity, bending stiffness, and cross-membrane chemical transport. The package essentially processes all computational workloads in parallel by GPU, and it incorporates multi-stream scheduling and non-blocking MPI communications to improve inter-node scalability. Our code is validated for accuracy and compared against the CPU counterpart for speed. Strong scaling and weak scaling are also presented to characterize scalability. We observe a speedup of 10.1 on one GPU over all 16 cores within a single node, and a weak scaling efficiency of 91% across 256 nodes. The program enables quick-turnaround and high-throughput numerical simulations for investigating chemical-driven red blood cell phenomena and disorders.

Program summary

Program Title: USERMESO 2.0

Program Files doi: http://dx.doi.org/10.17632/89849t3ngk.1

Licensing provisions: GNU General Public License, Version 3

Programming language: C/C++, CUDA C/C++, MPI.

Nature of problem: Particle-based simulation of a red blood cell suspension with chemical transport property.

Solution method: Each red blood cell is represented by a 3-D triangular mesh with bonded potential under area and volume constraints. The solvent is approximated with coarse-grained particles. The time evolution of the system is integrated using Velocity-Verlet algorithm.

Restrictions: The code is compatible with NVIDIA GPGPUs with compute capability 3.0 and above.

Unusual features: The code is implemented on GPGPUs with significantly improved speed.

Additional Comments: Github repository link https://github.com/AnselGitAccount/USERMESO-2.0

1. Introduction

Blood carries nutrients, hormones, and waste products around the body. Roughly 35–45% of its volume is occupied by red blood cells (RBCs) that are responsible for vital biological tasks such as circulating oxygen and carbon dioxide throughout the body. As illustrated in Fig. 1, the exchange of materials between blood and its surrounding tissues occurs primarily in the microvascular beds where arterioles and venules meet. The capillaries connecting the arterioles and venules are ideal for chemical diffusion due to their large surface-to-volume ratio and single-layered fenestrated vessel wall. However, this seemingly simple process is underpinned by intricate and detailed mechanisms. In the example of oxygen release from RBCs, the amount of oxygen discharged depends on the detailed chemical balancing between compounds

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding authors.
E-mail addresses: ansel_blumers@brown.edu (A.L. Blumers), zhen_li@brown.edu (Z. Li), george_karniadakis@brown.edu (G.E. Karniadakis).

http://dx.doi.org/10.1016/j.cpc.2017.03.016
0010-4655/

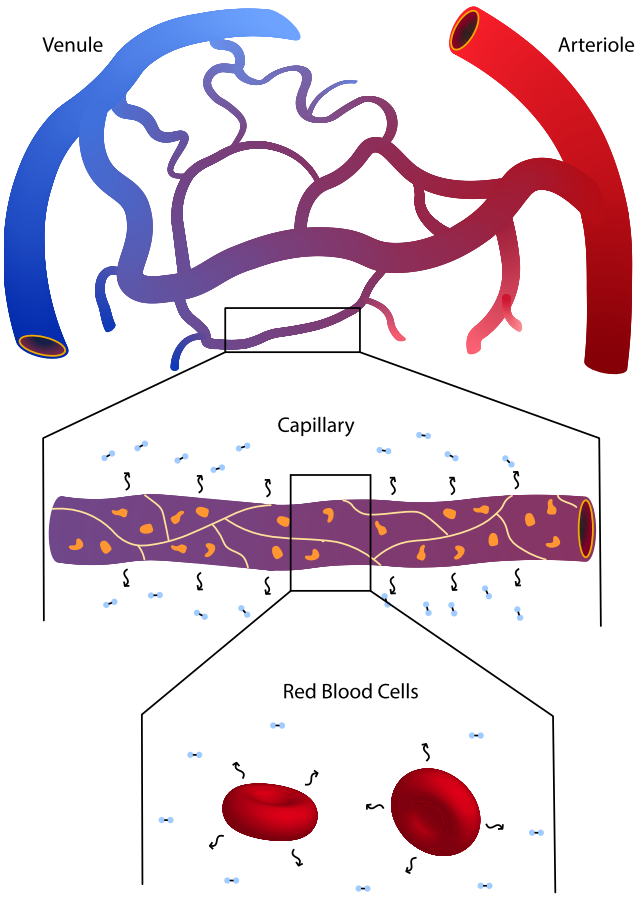

Fig. 1. An illustration of oxygen release in the microvascular bed. RBCs transverse from oxygen-rich arterioles to oxygen-scarce venules through the connecting capillary beds. In each segment of the fenestrated capillary, oxygen and other nutrients diffuse into the surrounding tissue through the porous wall. The oxygen is carried and released almost exclusively by hemoglobin contained in RBCs. Our long-term goal is to simulate the chemical exchange process with explicit modeling of the RBCs as sources at the mesoscopic level. The package presented in this work serves as an enabling technology to this objective.

such as hemoglobin and carbon dioxide, as well as ambient parameters such as plasma solubility and tissue permeability. Simulating the chemical-exchange process and capturing important details of blood flow can improve our understanding of the biological mechanisms. Such technology can be used to further our investigations into totally unexplored mechanisms linking quantitatively metabolomics with blood flow in a very precise way for the first time.

Simulations of chemical exchange in the microvascular beds with an explicit description of the source RBCs involve many biological phenomena at length scales from nanometers to micrometers. This is difficult to accomplish with either continuum descriptions based on partial differential equations or atomistic models based on classical Hamiltonian mechanics. Mesoscopic simulation methods such as Dissipative Particle Dynamics (DPD) [1] are gaining momentum as a promising approach to capture these phenomena at this intermediate scale. In contrast to Brownian dynamics and generalized Langevin dynamics, the pairwise force in DPD depends on the relative position and velocity between each pair of interacting particles. This ensures Galilean invariance and momentum conservation that allow the statistical recovery of the Navier–Stokes equation [2]. Consequently, DPD can correctly reproduce hydrodynamic behavior at the mesoscale. The versatility of DPD has been successfully demonstrated for many interesting biological applications, e.g., blood rheology [3], platelet aggregation [4], cell sickling [5], and polymer self-assembly [1,6,7]. Our newly developed transport DPD (tDPD) [8] can model the diffusion, advection, and reaction of the chemical transport process on top of the classic DPD framework at the molecular level using a Lagrangian framework.

Developing a realistic RBC membrane model is crucial because the RBC mechanical and rheological state influences how an RBC performs its biological function. A coarse-grained DPD membrane model for RBC was pioneered by Pivkin et al. [9]. It takes bending energy, in-plane shear energy, and area and volume constraints into consideration to recover the correct bending stiffness and shear modulus. Fedosov et al. [10] later extended the aforementioned model to capture the correct non-linear deformation of RBC under stress as measured from optical tweezers experiments. Although the membrane model has been carefully calibrated, large-scale RBC simulations using the DPD model remain rare, while existing works are primarily limited to use a few tens of RBCs at a time. The large computational overhead has severely limited the possibility for blood-flow studies in vascular networks using the DPD-based RBC model.

The effective use of General Purpose Graphics Processing Units (GPGPUs) has improved the capability of many molecular dynamics simulation software by an order of magnitude thanks to its massively parallel nature [11–18]. Pushing for the peak performance on each of the 18,688 GPUs on the Titan supercomputer, Tang et al. [19] were able to simulate billions of RBCs flowing through a cancer cell filtering device. Despite the inclusion of GPUs to accelerate the simulation and to broaden the accessible time and length scale, the code was a fixed-purpose solver based on the classic DPD model.

In this work, a generalized GPU-accelerated implementation of the RBC model with tDPD adaptation is developed to simulate RBCs with realistic physiological properties. The software features a tight integration of our earlier work on RBC membrane model, transport behavior simulations, and GPU-accelerated DPD simulators. With the new ability to track chemical concentrations, the program can be used to investigate chemical-driven or chemical-sensitive phenomena or disorders with unprecedented time and length scales. The rest of the paper is structured as follows. In Section 2, we review the classic DPD and tDPD frameworks and present a short summary of the RBC membrane model. In Section 3, we present our software implementations and algorithmic innovations. In Section 4, we validate the code with verification cases. In Section 5, we demonstrate the efficiency of the code as reflected by benchmark cases simulating pure tDPD fluids and RBC suspensions. In Section 6, we further demonstrate the capability of the software with a realistic microfluidic channel simulation. We conclude the paper in Section 7.

2. Mathematical model

2.1. tDPD pairwise interaction

The classic DPD framework describes the evolution of density and velocity fields through Newton's laws. In addition to the conservation of momentum inherited from the classic DPD, tDPD also conserves the concentrations of species. Aside from its position and velocity, each particle explicitly tracks the concentrations of all species in the volume represented by this particle. A concentration gradient induces the flux of chemicals between pairs of particles, in a fashion similar to heat transfer.

The movements of particles can be described by the following set of stochastic ordinary differential equations (ODEs) [20]:

$$\frac{d\mathbf{r}_i}{dr} = v_i, \tag{1}$$

$$\frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i = \sum_{i\neq j}(\mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R), \tag{2}$$

where t, $\mathbf{r}_i$, $\mathbf{v}_i$ and $\mathbf{F}_i$ denote time, position, velocity, and force, respectively. The net force imposed on particle i is the sum of conservative force $\mathbf{F}_{ij}^C$, dissipative force $\mathbf{F}_{ij}^D$, and corresponding random force $\mathbf{F}_{ij}^R$ via interactions with every particle j within a radial cutoff r_c of i. Those forces are given as [2]:

$$\mathbf{F}_{ij}^C = \alpha_{ij}\ \omega_C(r_{ij})\ \mathbf{e}_{ij}, \tag{3}$$

$$\mathbf{F}_{ij}^D = -\gamma_{ij}\ \omega_D(r_{ij})\ (\mathbf{e}_{ij}\cdot\mathbf{v}_{ij})\ \mathbf{e}_{ij}, \tag{4}$$

$$\mathbf{F}_{ij}^R = \sigma_{ij}\ \omega_R(r_{ij})\ \xi_{ij}\ \Delta t^{-1/2}\ \mathbf{e}_{ij}, \tag{5}$$

where $\mathbf{e}_{ij} = \mathbf{r}_{ij}/r_{ij}$ is the unit vector from particles i and j. Δt is the time step of the simulation, and ξ is a symmetric Gaussian random variable with zero mean and unit variance [2]. α_{ij}, γ_{ij}, and σ_{ij} are conservative, dissipative, and random force coefficients, respectively. $\omega_C(r_{ij})$, $\omega_D(r_{ij})$, and $\omega_R(r_{ij})$ are the corresponding weighting functions. The relationship between dissipative and random effects is dictated by the fluctuation–dissipation theorem which imposes the following constraints [21]:

$$\sigma_{ij}^2 = 2\,k_BT\,\gamma_{ij}, \qquad \omega_D(r_{ij}) = \omega_R^2(r_{ij}), \tag{6}$$

where k_B is the Boltzmann constant, and T is temperature.

The tDPD model enables us to capture reaction kinetics at the mesoscopic level. It essentially solves the advection–diffusion–reaction equation $\frac{dC}{dt} = D\nabla^2 C + Q^S$, where the transport of concentration is modeled by a Fickian flux and a random flux [22,23]. It can be described by the following equation:

$$\frac{dC_i}{dt} = Q_i = \sum_{i\neq j}(Q_{ij}^D + Q_{ij}^R) + Q_i^S, \tag{7}$$

where C_i denotes the concentration of a species carried by particle i, and Q_i is the net flux. There are three flux components that can potentially alter the concentration. Q_{ij}^D and Q_{ij}^R denote Fickian and random fluxes respectively and are given by [8]

$$Q_{ij}^D = -\kappa_{ij}\ \omega_{DC}(r_{ij})\ (C_i - C_j), \tag{8}$$

$$Q_{ij}^R = \epsilon_{ij}\ \omega_{RC}(r_{ij})\ \Delta t^{-1/2}\ \zeta_{ij}, \tag{9}$$

where κ_{ij} and ϵ_{ij} adjust the strengths. The corresponding weights $\omega_{DC}(r_{ij})$ and $\omega_{RC}(r_{ij})$ are given as $\omega_{DC} = (1-r/r_{cc})^{s_{c1}}$ and $\omega_{RC} = (1-r/r_{cc})^{s_{c2}}$ with a cutoff radius r_{cc}, analogous to the weights in force calculations. Taking the same idea from the fluctuation–dissipation theorem in the classic DPD, the transport version requires [24]

$$\epsilon_{ij}^2 = m_s^2\,\kappa_{ij}\,\rho\,(C_i + C_j), \qquad \omega_{DC}(r_{ij}) = \omega_{RC}^2(r_{ij}), \tag{10}$$

where ρ is the tDPD particle density.

It is easy to see the parallel between Eqs. (6) and (10). In particular, parameter κ adjusts the strength of chemical transfer, as the dissipative coefficient γ adjusts the strength of momentum dissipation. It is worth noting that each species typically has a distinct κ value because diffusion coefficients for different chemical species are generally different. Since the mass of a single solute molecule m_s is much smaller than the mass of a tDPD particle m which is usually chosen to be one (normalized value), the magnitude of ϵ_{ij} is insignificant. This translates to a negligible contribution from Q_{ij}^R to the diffusion coefficient. In this work, Q_{ij}^R can be safely ignored considering the simplification $m_s \ll m$ [8]. Finally, the source term Q_i^S represents the external contributions e.g., external concentration source and boundary conditions.

The random force component in the classic DPD is stochastic by definition. The stochastic behavior of particle movement yields an additional contribution to diffusion reflected in the diffusion coefficient. This additional diffusion component D^ζ and the Fickian diffusion component D^F compose the effective diffusion coefficient D. Because the random flux Q^R is insignificant, D^ζ is due to random movement of particles solely and can be calculated by [2]

$$D^\zeta = \frac{3k_BT}{4\pi\gamma\rho\cdot\int_0^{r_c} r^2\omega_D(r)g(r)dr}. \tag{11}$$

The Fickian concentration flux Q_{ij}^D depends on D^F entirely, which can be calculated by [8]

$$D^F = \frac{2\pi\kappa\rho}{3}\int_0^{r_{cc}} r^4\omega_{DC}(r)g(r)dr. \tag{12}$$

The physical diffusion coefficients are mapped to those of the particle simulation through parameter κ. The mapping relation can be extracted from matching simulation results with known analytical solutions as described in [8].

2.2. RBC model

RBC membrane comprises a lipid bilayer supported by an inner cytoskeleton. Composed of spectrin proteins and actins in a compact network, the cytoskeleton provides structural stability. Membrane viscosity and elasticity, as well as bending stiffness, are physical properties derived from these biological components. The same physical properties can be recovered with a spring-network model that resembles a triangular mesh on a 2D surface as described in [10].

The shape of the viscoelastic membrane is maintained by a potential derived from a combination of bond, angle and dihedral interactions. The bonds, also referred to as springs, experience both an attractive and a repulsive component. The attractive potential adopts the form of the wormlike chain potential and is given by

$$U_{WLC} = \frac{k_BTh_m}{4p}\frac{\frac{h}{h_m}^2(3-2\frac{h}{h_m})}{1-\frac{h}{h_m}}, \tag{13}$$

where k_BT is the energy per unit mass, h_m is the maximum spring extension, and p is the persistence length. The repulsive potential adopts the form of a power function given by

$$U_{POW} = \begin{cases} \dfrac{k_p}{(m-1)h^{m-1}}, & m\neq 1, \\ -k_p\log(h), & m = 1, \end{cases} \tag{14}$$

where m is positive exponent, and k_p is force coefficient. In addition to those conservative potentials, a viscous component is needed to damp the springs. It is realized by a dissipative force, as well as the corresponding random force, given as

$$\mathbf{F}_{ij}^D = -\gamma^T\mathbf{v}_{ij} - \gamma^C(\mathbf{v}_{ij}\cdot\mathbf{e}_{ij})\mathbf{e}_{ij}, \tag{15}$$

$$\mathbf{F}_{ij}^R = \sqrt{2k_BT}\left(\sqrt{2\gamma^T}(d\mathbf{W}_{ij}^S - tr[d\mathbf{W}_{ij}^S]\mathbf{I}/3) + \frac{\sqrt{3\gamma^C-\gamma^T}}{3}tr[d\mathbf{W}_{ij}]\mathbf{I}\right), \tag{16}$$

where γ^T and γ^C are dissipative coefficients, and v_{ij} is the relative velocity. $d\mathbf{W}_{ij}^S$ is the symmetric component of a random matrix of independent Wiener increments $d\mathbf{W}_{ij}$.

The angle interaction is facilitated by area and volume constraints given by

$$V_{area} = \frac{k_g(A^t - A_0^t)^2}{2A_0^t} + \sum_j \frac{k_l(A_j - A_{0,j})^2}{2A_{0,j}}, \tag{17}$$

$$V_{volume} = \frac{k_v(V^t - V_0^t)^2}{2V_0^t}, \tag{18}$$

where j is triangle index. k_g, k_l, and k_v are global area, local area, and global volume constraints coefficients, respectively. The instantaneous area and volume of an RBC are denoted by A^t and V^t, whereas A_0^t and V_0^t represent the equilibrium area and volume. A_0^t is calculated by summing the area of each triangle. V_0^t is found according to scaling relationship $V_0^t/(A_0^t)^{3/2} = V^R/(A^R)^{3/2}$, where V^R and A^R are the experimental measurements.

The dihedral interaction is described by the bending potential given by

$$V_{bending} = \sum_j k_b\Big(1 - \cos(\theta_j - \theta_0)\Big), \tag{19}$$

where k_b is the bending constant, and θ_j is the angle between two neighboring triangles with the common edge j. More detailed information regarding RBC membrane model can be found in [10].

3. Parallel implementation

3.1. USER-MESO

USER-MESO [25] is a GPU-accelerated extension package to LAMMPS [16] for dissipative particle dynamics and smoothed particle hydrodynamics simulations. It migrates all computation workload, except inter-rank communications, onto GPUs and achieves more than twenty times speedup on a single GPU over 8–16 CPU cores. It utilizes MPI for the inter-node communication and can scale to at least 1024 nodes. The list below summarizes the major technical innovations involved in implementing the package:

- An atomics-free warp-synchronous neighbor list construction algorithm;
- A 2-level particle reordering scheme, which aligns with the cell list lattice boundaries for generating strictly monotonic neighbor list;
- A locally transposed neighbor list;
- Redesigned non-branching transcendental functions (sin, cos, pow, log, exp, etc.);
- Overlapping pairwise force evaluation with halo exchange using CUDA streams for hiding the communication and the kernel launch latency;
- Radix sort with GPU stream support;
- Pairwise random number generation based on binary particle signatures and the Tiny Encryption Algorithm.

3.2. Data layout

Data in LAMMPS are stored in array of structure layout. To avoid strided access on the GPU, USER-MESO employs structure of array layout on GPU instead. A pair of interleave/deinterleave kernels facilitates the conversion between array of structure and structure of array at runtime. Meanwhile, the concentration and flux arrays of a particle must be able to hold an arbitrary number of species. Accessing this information is more efficient when the data are coalesced. Structure of Array layout is therefore chosen for storing concentrations and fluxes in CPU and Array of Structure layout in GPU.

3.3. RBC

Three bonds form a triangle in the 2-D triangulated surface. Each triangle is associated with an area and a "quasi"-volume that is calculated from the absolute distances of three vertexes. Because thread-to-thread communication is expensive, it is advantageous to compute area and volume three times, one by each thread. The excessive computation outweighs thread-to-thread communication overhead.

The angle term demands the most computational resources because the total area and volume for each RBC need to be calculated before enforcing the area and volume constraints. This is time-consuming especially when an RBC is split between two MPI ranks. Although the areas and volumes are accumulated in a rank-basis for the portion of an RBC in that rank, the total areas and volumes must be known for each and every rank that contain any portion of that RBC. This procedure can be accomplished with an MPI-allreduce between two kernel calls for each time step. The first kernel, *K-Gather*, computes the total area and volume of each RBC pertaining to that particular rank. The second kernel, *K-Apply*, then enforces the area and volume constraints.

This sequential procedure offers no computation overlap between those two GPU kernels because *K-Apply* awaits the result of *K-Gather*. However, when the time step is very small, the variation in area and volume between two consecutive steps is insignificant. Adopting area and volume from the previous time step in *K-Apply* for the current time step permits concurrency between *K-Apply* and MPI communication, thus overlapping CPU and GPU tasks for efficiency. A comparison with the sequential version shows visually identical results in Fig. 2.

Since *K-Gather* and *K-Apply* are independent procedures in the concurrent version, computing *K-Gather* and *K-Apply* simultaneously is possible due to CUDA stream functionality. Instead of occupying the default stream exclusively, they are assigned to two streams as demonstrated in Fig. 3. CUDA-events are placed accordingly for synchronization. Opposite to what we expect, this setup produces worse performance. This can be explained by streaming multiprocessor saturation — two kernels competing for computing resources. The resulting higher cache refresh rate is detrimental to efficiency. Our hypothesis is validated by the fact that each kernel takes longer than its sequential counterpart to complete.

Even though concurrent kernels setup produces undesirable performance, it prompts the possibility of simultaneous data transfer and kernel execution, as well as the possibility of non-blocking MPI communications. MPI communications are blocking by default. An imbalanced workload on each device exacerbates the latency because blocking MPI communications can only initiate when all devices synchronize with the CPUs. The more ranks there are, the bigger potential latency there is. Non-blocking MPI communication is generally preferred for this reason. The pseudocode of an improved implementation is depicted in Algorithm 1, and a graphical representation is illustrated in Fig. 4. In the improved version, MPI-Wait triggers Memcpy-HtD that uploads area and volume from the previous time step from CPU to GPU. Because the upload and *K-Gather* execution share no dependency, Memcpy-HtD can be implemented in a different stream for efficiency. *K-Apply* execution starts as soon as the result, area and volume from current time step, from *K-Gather* are downloaded to CPU via Memcpy-DtH. The completion of Memcpy-DtH also triggers the non-blocking MPI communication as a concurrent process. The non-blocking MPI allows the subsequent kernel or CPU processes to continue prior to the *K-Apply* completion. Communication overhead is thus greatly reduced. This setup maximizes GPU efficiency by eliminating SM unoccupancy and hiding default MPI communication overhead.

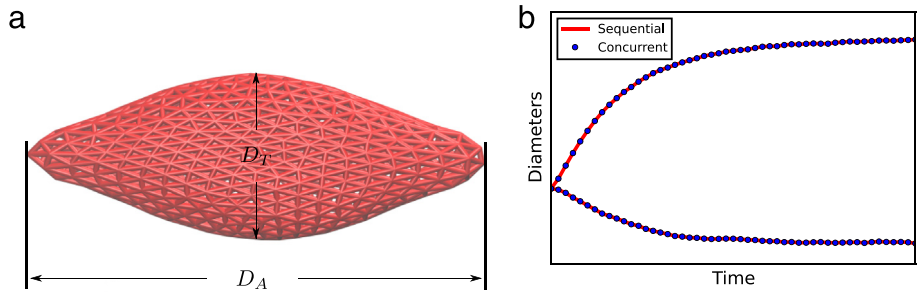

Fig. 2. An RBC stretching test was devised to show feasibility of adopting area and volume of previous time step for current time step. (a) A graphical representation of RBC stretching is shown. D_T and D_A denote transversal and axial diameters, respectively. (b) The time evolutions of D_T (top curve) and axial diameter D_A (bottom curve) are shown. The simulation results from the concurrent version match those of the sequential version.

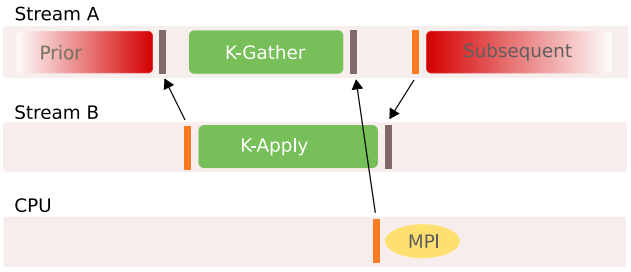

Fig. 3. The multi-stream scheduling schematic drawing. The arrows denote dependency. *K-Apply* has to wait until the CUDA-event marking the completion of *K-Gather* is recorded. The same applies to the MPI communication.

Algorithm 1 Pseudocode to the concurrent implementation.

1: Wait for the completion of **MPI-Iallreduce** from last time step.
2: Upload data to device with asynchronous **Memcpy-HtD**.
3: Compute the total area and volume of each RBC in **K-Gather**.
4: Place asynchronous **Memcpy-DtH** in execution queue.
5: Download data to host with asynchronous **Memcpy-HtD**.
6: Enforce the area and volume constraints in **K-Apply**.
7: Wait for the completion of **Memcpy-DtH**.
8: Sum total area and volume with non-blocking **MPI-Iallreduce**.

4. Code validation

4.1. Flow

Force accuracy over long-time integration is validated by simulating a transient double Poiseuille flow. The parallel flow is driven by a body force f on tDPD particles. The system consists of 256,000 particles in a $40 \times 40 \times 40$ box with periodic boundary conditions in all directions. The simulation parameters were chosen as $\alpha = 18.75$, $\gamma = 4.5$, $r_c = 1.58$, and $s = 0.41$. Results shown in Fig. 5 are in good agreement with the analytical solution in [26].

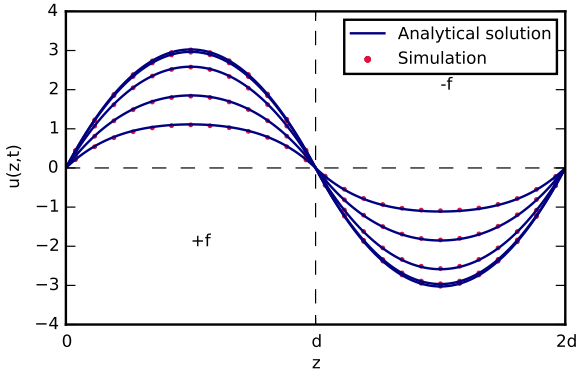

Fig. 5. Transient velocity profile in a double Poiseuille flow. A body force f was imposed from 0 to d, and a body force $-f$ was imposed from d to $2d$. The boundary in z direction is thus zero by periodicity. Snapshots are taken at $t = 3, 6, 12, 24$ and 48, corresponding to the curves from bottom to top, to show the evolution of velocity profiles.

4.2. Transport

The diffusive property is validated by solving a one-dimensional diffusion equation in an infinitely long cylinder with circular cross-section. The cylinder with radius $R = 20$ is composed of 402,073 particles in a $80 \times 40 \times 40$ domain. The simulation parameters associated with force calculation were chosen as $\alpha = 18.75$, $\gamma = 4.5$, $r_c = 1.58$, and $s = 0.41$. For flux calculations, the simulation parameters were chosen as $s_2 = 2.0$, $\kappa = 5.0$, and $r_{cc} = 1.58$ as explained in Section 2.1. An initial uniform concentration C_0 and constant Dirichlet boundary C_D were implemented. The Dirichlet boundary condition is realized with the effective boundary method demonstrated in [8]. This method also satisfies the no-slip boundary condition [5,27].

The exact solution $C(t)$ to the diffusion equation $\frac{\partial C}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}(rD\frac{\partial C}{\partial r})$ is given as [28]

$$\frac{C(t) - C_0}{C_D - C_0} = 1 - \frac{2}{R}\sum^{\infty} \frac{J_0(\lambda_n r)}{\lambda J_1(\lambda_n R)} \cdot \exp(-D\lambda^2 t), \tag{20}$$

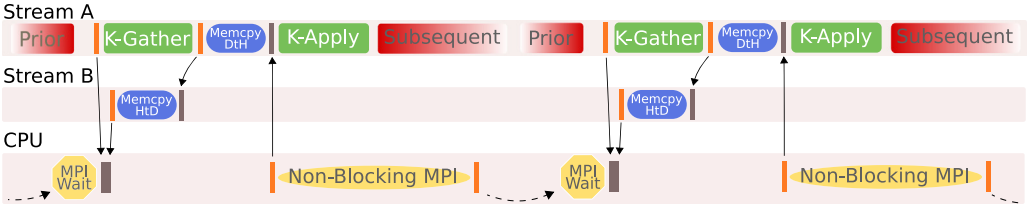

Fig. 4. The solid arrows denote dependency within a time step, and the dashed arrows denote dependency across one time step. Async-Memcpys and Kernels are set up concurrently whenever possible to engage compute and copy engines simultaneously. Non-Blocking MPI eliminates the need for an additional device synchronization. See file *angle_area_volumn_meso.cu*, lines *593–704*.

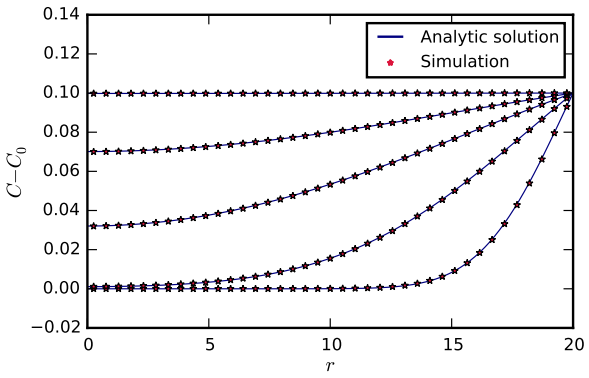

Fig. 6. Time evolution of concentration profiles in an axial symmetric infinitely long tube. Initially the system has concentration C_0 everywhere. The boundary is kept at $C_0 + 0.1$ for $t > 0$. Snapshots are taken at $t = 1, 5, 15, 30$ and 100 to show the evolution, from the bottom curve to the top.

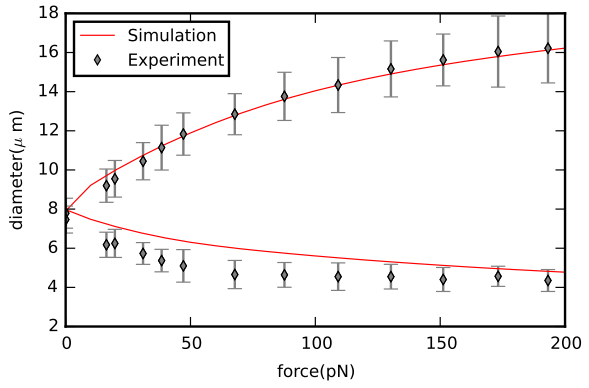

Fig. 7. RBC elasticity validation. The top curve represents axial diameter D_A, and the bottom curve represents transverse diameter D_T. As Fedosov et al. [10] pointed out, the small discrepancy in transverse diameter between simulation and experiment is probably due to the optical measurement being performed from only one angle.

where R is the radius, and λ_n are the positive roots of Bessel functions of the first kind $J_0(R\lambda_n) = 0$. The simulation result matches the analytical solution as shown in Fig. 6.

4.3. RBC elasticity

For validation, the RBC implementation was compared with experimental data [29] from an RBC stretching experiment via optical tweezers. The stretching was simulated by applying a constant force on few particles in opposing ends of an RBC [10]. By varying the stretching force, the axial diameter D_A and transverse diameter D_T adjust according to the membrane elasticity. Excellent matching between the simulation and the experiment was obtained, as shown in Fig. 7. The result is identical to the one obtained using a different code by Fedosov et al. [10].

5. Benchmark

5.1. Method

The benchmarks were run on Titan at the Oak Ridge National Laboratory. Titan is a Cray XK7 system employing 18,688 nodes, each containing an AMD *Opteron* 6274 CPU and a NVIDIA *Tesla* K20X GPU. Every CPU contains 16 integer cores and 8 floating point cores clocked at 2.2 GHz. The Kepler architecture GPU has 2688 CUDA cores with a peak double-precision performance of

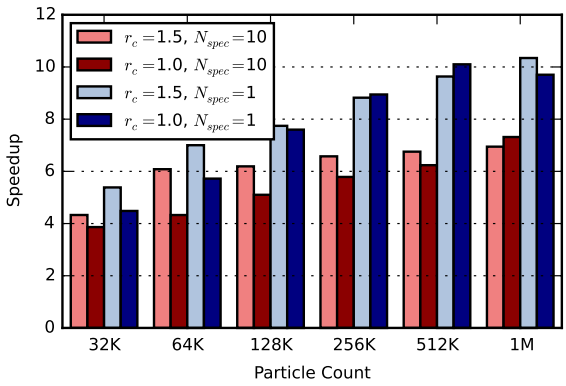

Fig. 8. Speedup of pure tDPD fluid simulation over the CPU solver. Two different neighbor list cutoff values, $r_c = 1.0$ and 1.5, as well as two different numbers of chemical species, $N_{spec} = 1$ and 10, were considered. Neighbor lists were updated every 5 steps.

Table 1
Speedup of RBC suspension simulation over the CPU solver. The particle count in a domain depends on *Hct*. For example, *Hct* 7% for the system volume of 8192 translates to six RBCs. Combining with 32,768 pure fluid particles, this RBC suspension system has 35,768 tDPD particles total.

Hct	System volume	Solvent particles	RBC count	Total Particle count	Speedup
7%	8,192	32,768	6	35,768	3.8
	16,384	65,536	12	71,536	5.1
	32,768	131,072	24	143,072	5.4
	65,536	262,144	49	286,644	5.7
35%	8,192	32,768	30	47,768	4.5
	16,384	65,536	61	96,036	5.3
	32,768	131,072	123	192,572	5.9
	65,536	262,144	246	385,144	6.7

1.31 TFLOPS. The CPU solver is compiled with GCC, $-O3$ optimization. The GPU version uses NVIDIA NVCC compiler with $-O3$ optimization.

Two system setups, *i.e.* pure tDPD fluid and RBC suspension, were tested. The pure tDPD fluid system consists of simple tDPD particles with pair-interaction only, whereas the system of RBC suspension contains a combination of pure tDPD fluid and coarse-grained RBCs. For a fair comparison, only the main execution loop was timed.

5.2. Single node speedup

Pure tDPD fluid

The key performance metric used in this work is speedup, defined as the ratio of time elapsed between the CPU and GPU implementations. The benchmark reveals significant speedup up to 10.3 times over the CPU solver for different parameter values as shown in Fig. 8. When a large number of species is included, the performance declines because the computation becomes memory-bound. Although N_{spec} is an impacting factor, the GPU version still delivers up to 7.2 times speedup even at $N_{spec} = 10$. Solutions such as reduced precision or data compression are viable to alleviate the bottleneck.

RBC suspension

Systems with two different Hematocrit (*Hct*), 7% and 35%, were benchmarked. Each RBC is represented by 500 tDPD particles. The number of total tDPD particles is the sum of tDPD solvent particles and RBC particles (Table 1). In this case, only one species was included. The speedup is comparable to pure tDPD fluid of similar total particle count.

Table 2
Speedup of RBC suspension simulation on TITAN X Maxwell and Pascal over the CPU solver. Under the same simulation domain setup with Hct 35%, the GPUs with newer architectures outperform the K20X (Kepler) that is currently employed at Oak Ridge National Laboratory. The larger speedup is contributed by the increased bandwidth and higher top speed.

System volume	Total particle count	TITAN X Maxwell speedup	TITAN X Pascal speedup
8,192	47,768	4.8	6.5
16,384	96,036	5.8	9.2
32,768	192,572	7.2	9.9
65,536	385,144	7.2	10.1

We also tested the performance of the code on newer GPU architectures, namely Maxwell and Pascal (Table 2). The workstation used for benchmark is equipped with two Intel Xeon E5-2630L CPUs at 2.0 GHz, one GeForce TITAN X Maxwell GPU, and one GeForce TITAN X Pascal GPU. The speedup is measured as the ratio between the simulation wall time with 1 GPU or 8 CPU cores in the workstation.

5.3. Weak & strong scalings

The same neighbor list update frequency and r_c from the single node benchmarks were used to establish the weak scaling and strong scaling benchmarks. The metric (million particles) · (steps per second), or *MPS/second*, is used for absolute performance characterization and comparison across different systems.

Pure tDPD fluid

Weak and strong scalings were benchmarked on up to 1024 nodes. With 524,288 particles per node, weak scaling demonstrates nearly linear scaling. Strong scaling with a fixed system size of 2,097,152 particles is also presented in Fig. 9. Absolute performance plateaus around 512 nodes, where the parallel overhead cancels any extra computational resources. At 1024 nodes, the parallel overhead and data transfer completely dominate the computation.

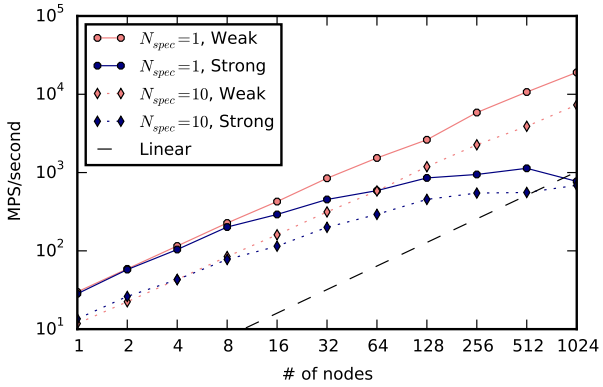

Fig. 9. Weak scaling and strong scaling of pure tDPD fluid particles in a log–log plot.

RBC suspension

Strong scaling with a system volume of 2,097,152 is shown in Fig. 10. It is clearly seen that the non-blocking implementation of the angle term is more efficient at large node counts. The slowly diverging absolute performance plots reveal the increasing penalty caused by blocking MPI communications. Switching to non-blocking reduces the execution time by approximately 25% in the case of 1024 nodes. Weak scaling of system volume of 32,768 per node also shows nearly linear scalability. The non-blocking version clearly delivers a better scalability.

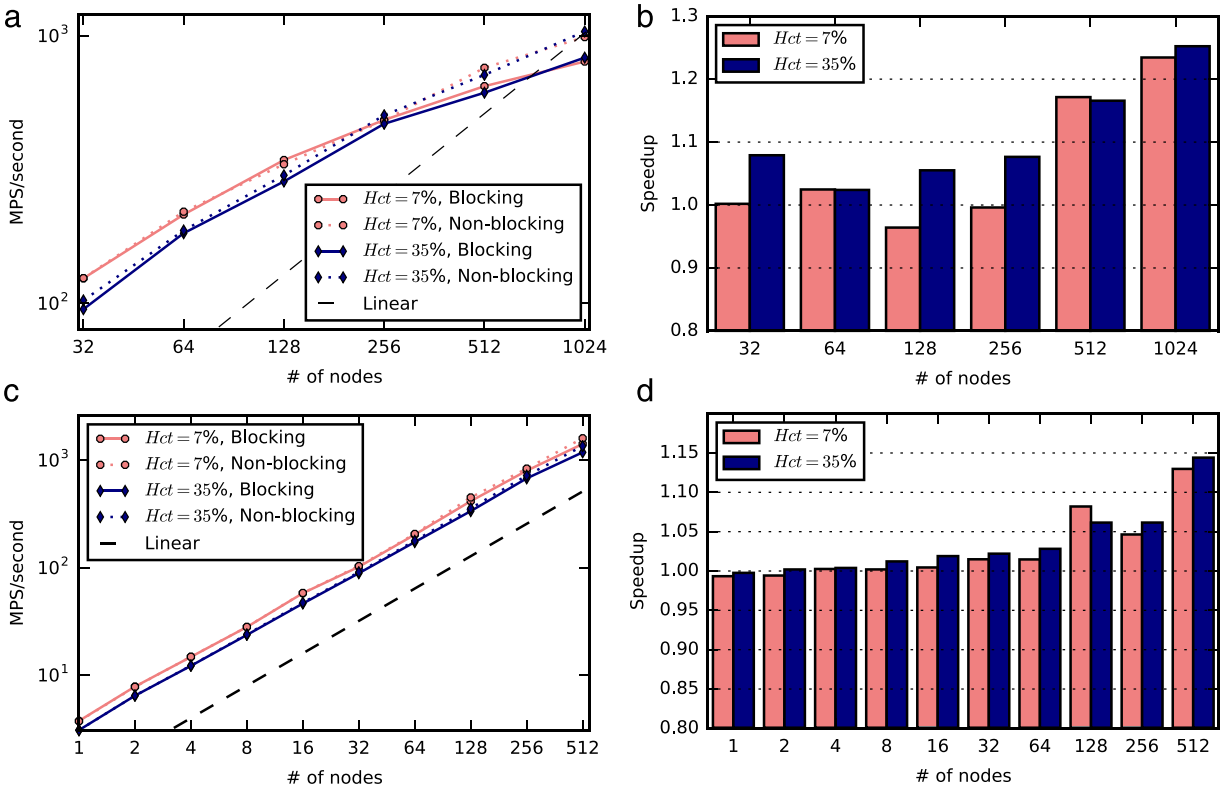

Fig. 10. Log–log plots of (a) strong scaling and (c) weak scaling of the RBC suspension system. The speedup between non-blocking and blocking implementations for (b) strong and (d) weak scalings. Compared to the blocking versions, the non-blocking counterparts show better scalability at high node counts.

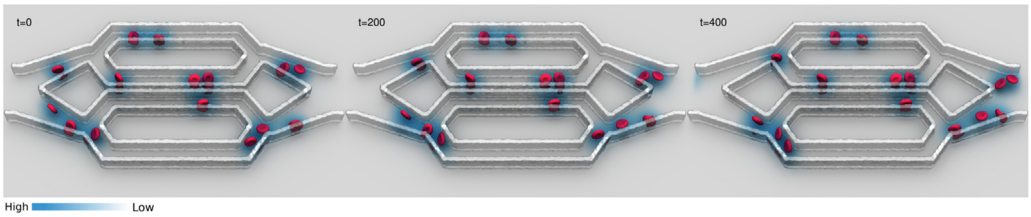

Fig. 11. An example simulation combining RBC model and tDPD formulation is shown here. The RBC–fluid mixture flows through a microfluidic device from left to right. The blue intensity field represents concentration gradient qualitatively. The chemical released by RBCs diffuses and dissipates in the fluid. The snapshots were taken at time 0, 200, and 400. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Capability demonstration

The role of RBCs in microcirculation has been investigated using microfluidic devices [30–32]. However, certain properties such as chemical concentrations are difficult to capture experimentally with high spatial and temporal resolution. The technology presented in this work can capture the concentration fields of an arbitrary number of species in one simulation.

To demonstrate versatility of our software, chemical diffusion from an RBC suspension flowing in a microfluidic device was simulated. As the chemical diffuses from the RBCs, it reacts with the solvent and dissipates. In reality, the dissipation can be induced by a number of causes, such as chemical reaction, disintegration and evaporation. The blue intensity field shown in Fig. 11 represents concentration gradient qualitatively. The bluish tone surrounding RBCs indicates abundant presence of the chemical near the sources.

Among the 720,778 tDPD particles in the simulation, roughly 95% represents the fluid and the solid channel wall at a density of 4 particles per unit volume. For each RBC, the membrane is composed of 500 particles connected by a network of springs, and the cytoplasm is portrayed by 372 particles enclosed by the membrane. The cytoplasm and fluid particles reside in their respective sides, enforced by a novel boundary resolving technique [33]. This technique is also applied on the fluid–wall interface to prevent penetration of fluid particles. The same simulation setup will take approximately 12 times as long on the CPU, deduced from the benchmark results.

7. Summary

In this paper, a GPU-accelerated RBC simulation package based on a tDPD adaptation of our RBC model [10] is presented. Taking advantage of the ability to model chemical transport in tDPD, RBC dynamics and the advection–diffusion–reaction processes can be simulated simultaneously. The effective use of GPUs improves the code performance dramatically as revealed by the benchmarks done on Titan at Oak Ridge National Laboratory. With some novel algorithms in streamlining the RBC computation, our code was able to produce up to 10.1 times speedup when compared with its CPU counterpart on a single-node. The weak scaling benchmarks show almost linear scaling, while further speedup is possible even beyond 1024 nodes as indicated by strong scaling benchmarks. Furthermore, we demonstrated the software's capability by simulating chemical diffusion in an RBC suspension traversing inside a microfluidic device. Incorporating the boundary resolving technique that deals with arbitrary shapes [33], the software can easily reconstruct complex experimental apparatuses and perform realistic RBC simulations.

It should be stressed that the software presented in this study is highly customizable, as opposed to the fixed-purpose program by Rossinelli et al. [19]. We encourage researchers to adopt our user-friendly software in their research. The software is freely available on Github, following the link https://github.com/AnselGitAccount/USERMESO-2.0.

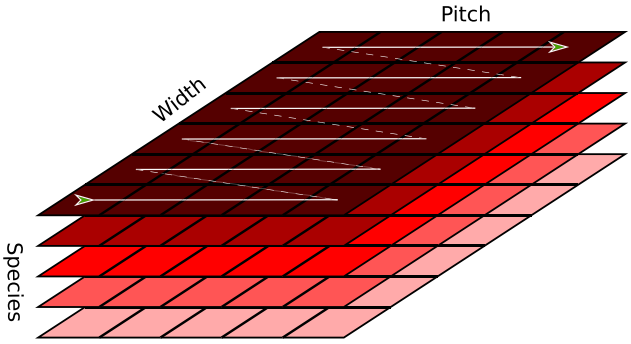

Fig. A.1. Schematic representation of 2D layered texture. Each layer holds the concentration of all particles for a particular species. The particles then wrap around to form a 2D array within each layer.

Acknowledgments

This work was supported by NIH grants U01HL114476 and U01HL116323. It was also sponsored by the US Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-12-2-0023. The benchmarks were performed on Titan at Oak Ridge National Laboratory through the Innovative and Novel Computational Impact on Theory and Experiment program under project BIP118. A.L.B. would like to acknowledge Wayne Joubert (Oak Ridge National Laboratory) for his effort to coordinate machine reservations. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce TITAN X Pascal GPU used for this research.

Appendix A

Another attempt in speeding up the program was to reduce concentration access-time on the GPU. Layered textures are commonly used due to their optimization on accessing data with spatial locality. Because 1D layered texture has a valid extent upper limit of 16,384 for Maxwell GPU architecture, 2D layered texture illustrated in Fig. A.1 must be used to hold a reasonable number of particles. Unexpectedly, the layered texture memory implementation performs worse than the non-texture version revealed by benchmarks. NVIDIA GPU profiling pinpoints "data request" as the main stall reason. The much diminished texture hit rate indicates that the limited texture cache resources are experiencing cache depletion. The data locality in coordinates and velocity texture cache are strategically optimized for cache hit rate. The concentration texture data deplete the cache and thus disrupt the data locality optimization. The overall texture cache hit rate is therefore reduced significantly, and this translates directly to worse performance.

Appendix B

Example simulations are included in the program source code package which can be downloaded from CPC or Github repository.

The instruction for compilation and execution is detailed in the top-level README file. A simple simulation of single RBC can be found under directory `<working_copy>/examples/simple`. Both the flow and transport validations described in Sections 4.1 and 4.2 are also included under directory `<working_copy>/examples/validations`, along with the optical tweezers simulation described in Section 4.3. A single-node benchmark described in Section 5.2 can be conducted by interested readers with files under directory `<working_copy>/examples/single_node_benchmark`.

References

[1] X. Li, Y.-H. Tang, H. Liang, G.E. Karniadakis, Chem. Commun. 50 (61) (2014) 8306–8308.
[2] R.D. Groot, P.B. Warren, J. Chem. Phys. 107 (11) (1997) 4423–4435.
[3] X. Li, Z. Peng, H. Lei, M. Dao, G.E. Karniadakis, Phil. Trans. R. Soc. A 372 (2021) (2014) 20130389.
[4] I. Pivkin, P. Richardson, G. Karniadakis, IEEE Eng. Med. Biol. Mag. 28 (2) (2009) 32–37.
[5] X. Li, P.M. Vlahovska, G.E. Karniadakis, Soft Matter 9 (1) (2013) 28–37.
[6] Z. Li, Y.-H. Tang, X.J. Li, G.E. Karniadakis, Chem. Commun. 51 (55) (2015) 11038–11040.
[7] Y.-H. Tang, Z. Li, X. Li, M. Deng, G.E. Karniadakis, Macromolecules 49 (7) (2016) 2895–2903.
[8] Z. Li, A. Yazdani, A. Tartakovsky, G.E. Karniadakis, J. Chem. Phys. 143 (1) (2015) 014101.
[9] I.V. Pivkin, G.E. Karniadakis, Phys. Rev. Lett. 101 (11) (2008) 118105.
[10] D.A. Fedosov, B. Caswell, G.E. Karniadakis, Biophys. J. 98 (10) (2010) 2215–2225.
[11] A.W. Goetz, M.J. Williamson, D. Xu, D. Poole, S.L. Grand, R.C. Walker, J. Chem. Theory Comput. 8 (2012) 1542–1555.
[12] B.R. Brooks, C.L. Brooks, A.D. Mackerell, L. Nilsson, R.J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, et al., J. Comput. Chem. 30 (10) (2009) 1545–1614.
[13] W. Smith, T.R. Forester, J. Mol. Graph. 14 (3) (1996) 136–141.
[14] H.J. Limbach, A. Arnold, B.A. Mann, C. Holm, Comput. Phys. Comm. 174 (9) (2006) 704–727.
[15] J.A. Anderson, C.D. Lorenz, A. Travesset, J. Comput. Phys. 227 (10) (2008) 5342–5359.
[16] S. Plimpton, J. Comput. Phys. 117 (1) (1995) 1–19.
[17] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, K. Schulten, J. Comput. Phys. 151 (1) (1999) 283–312.
[18] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M.R. Shirts, J.C. Smith, P.M. Kasson, D. Van Der Spoel, B. Hess, E. Lindahl, Bioinformatics 29 (7) (2013) 845–854.
[19] D. Rossinelli, Y.-H. Tang, K. Lykov, D. Alexeev, M. Bernaschi, P. Hadjidoukas, M. Bisson, W. Joubert, C. Conti, G. Karniadakis, M. Fatica, I. Pivkin, P. Koumoutsakos, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, pp. 1–12.
[20] P.J. Hoogerbrugge, J.M.V.A. Koelman, Europhys. Lett 19 (1) (1992) 155–160.
[21] P. Español, P. Warren, Europhys. Lett. 30 (4) (1995) 191–196.
[22] Z. Xu, P. Meakin, A. Tartakovsky, T.D. Scheibe, Phys. Rev. E 83 (6) (2011) 066702.
[23] J. Kordilla, W. Pan, A. Tartakovsky, J. Chem. Phys. 141 (22) (2014) 224112.
[24] J. Ortiz de Zárate, J. Sengers, Hydrodynamic Fluctuations in Fluids and Fluid Mixtures, Elsevier Science, Amsterdam, 2006.
[25] Y.-H. Tang, G.E. Karniadakis, Comput. Phys. Comm. 185 (11) (2014) 2809–2822.
[26] L.D.G. Sigalotti, J. Klapp, E. Sira, Y. Meleán, A. Hasmy, J. Comput. Phys. 191 (2) (2003) 622–638.
[27] Z. Li, Y.-H. Tang, H. Lei, B. Caswell, G.E. Karniadakis, J. Comput. Phys. 265 (2014) 113–127.
[28] J. Crank, The Mathematics of Diffusion, second revise ed., Oxford University Press, London, 1980, p. 73.
[29] S. Suresh, J. Spatz, J.P. Mills, A. Micoulet, M. Dao, C.T. Lim, M. Beil, T. Seufferlein, Acta Biomater. 23 (S) (2015) S3–S15.
[30] S.H. Holm, J.P. Beech, M.P. Barrett, J.O. Tegenfeldt, Lab Chip 11 (7) (2011) 1326–1332.
[31] R.D. Jäggi, R. Sandoz, C.S. Effenhauser, Microfluid. Nanofluid. 3 (1) (2007) 47–53.
[32] A.A.S. Bhagat, H.W. Hou, L.D. Li, C.T. Lim, J. Han, Lab Chip 11 (11) (2011) 1870–1878.
[33] Z. Li, X. Bian, Y.-H. Tang, G.E. Karniadakis, 2016, arXiv preprint arXiv:1612.08761.