Multiscale parareal algorithm for long-time mesoscopic simulations of microvascular blood flow in zebrafish

Ansel L. Blumers · Minglang Yin · Hiroyuki Nakajima · Yosuke Hasegawa · Zhen Li · George Em Karniadakis

Received: 26 January 2021 / Accepted: 1 July 2021 / Published online: 17 August 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
Various biological processes such as transport of oxygen and nutrients, thrombus formation, vascular angiogenesis and remodeling are related to cellular/subcellular level biological processes, where mesoscopic simulations resolving detailed cell dynamics provide a key to understanding and identifying the cellular basis of disease. However, the intrinsic stochastic effects can play an important role in mesoscopic processes, while the time step allowed in a mesoscopic simulation is restricted by rapid cellular/subcellular dynamic processes. These challenges significantly limit the timescale that can be reached by mesoscopic simulations even with high-performance computing. To break this bottleneck and achieve a biologically meaningful timescale, we propose a multiscale parareal algorithm in which a continuum-based solver supervises a mesoscopic simulation in the time-domain. Using an iterative prediction-correction strategy, the parallel-in-time mesoscopic simulation supervised by its continuum-based counterpart can converge fast. The effectiveness of the proposed method is first verified in a time-dependent flow with a sinusoidal flowrate through a Y-shaped bifurcation channel. The results show that the supervised mesoscopic simulations of both Newtonian fluids and non-Newtonian bloods converge to reference solutions after a few iterations. Physical quantities of interest including velocity, wall shear stress and flowrate are computed to compare against those of reference solutions, showing a less than 1% relative error on flowrate in the Newtonian flow and a less than 3% relative error in the non-Newtonian blood flow. The proposed method is then applied to a large-scale mesoscopic simulation of microvessel blood flow in a zebrafish hindbrain for temporal acceleration. The three-dimensional geometry of the vasculature is constructed directly from the images of live zebrafish under a confocal microscope, resulting in a complex vascular network with 95 branches and 57 bifurcations. The time-dependent blood flow from heartbeats in this realistic vascular network of zebrafish hindbrain is simulated using dissipative particle dynamics as the mesoscopic model, which is supervised by a one-dimensional blood flow model (continuum-based model) in multiple temporal sub-domains. The computational analysis shows that the resulting microvessel blood flow converges to the reference solution after only two iterations. The proposed method is suitable for long-time mesoscopic simulations with complex fluids and geometries. It can be readily combined with classical spatial decomposition for further acceleration.

Keywords Multiscale modeling · Zebrafish · Vascular network · Parallel-in-time · Dissipative particle dynamics · 1D blood flow modeling

Ansel L. Blumers and Minglang Yin have contributed equally to this work.

Minglang Yin
minglang_yin@brown.edu
Ansel L. Blumers
ansel_blumers@alumni.brown.edu
Yosuke Hasegawa
ysk@iis.u-tokyo.ac.jp
Zhen Li
zli7@clemson.edu
George Em Karniadakis
gorge_karniadakis@brown.edu

1 Department of Physics, Brown University, Providence, RI 02912, USA
2 Center for Biomedical Engineering, Brown University, Providence, RI 02912, USA
3 Department of Cell Biology, National Cerebral and Cardiovascular Center Research Institute, Osaka, Japan
4 Institute of Industrial Science, The University of Tokyo, Tokyo, Japan
5 Department of Mechanical Engineering, Clemson University, Clemson, SC 29634, USA
6 Division of Applied Mathematics, Brown University, Providence, RI 02912, USA


1 Introduction

Mesoscopic simulations are becoming increasingly important on studying material properties and interesting phenomena at the mesoscale, i.e., in biological and physical systems, thanks to its ability to reveal detailed dynamics with flexible timescales. As the spatiotemporal scales become small, the intrinsic stochastic effects can play important role in mesoscopic processes, which makes the continuum hypothesis become questionable. Instead, the mesoscopic system dynamics is suitably described by Lagrangian particle methods based on atomistic and molecular models [9,17,29]. Liu et al. [27] developed an immersed boundary method for finite elements that couples Navier–Stokes equations with cell interactions to capture important phenomena in blood rheology. More recently, many insightful works used in-silico models to advance the understanding of biological systems in migration/adhesion of tumor cells [46], asymmetry of cell-free layers in a in-vivo-like microvasculature, and red blood cells in the coronary artery [32] etc. Among them, Dissipative particle dynamics (DPD), as one of the most promising models in mesoscale simulations, is a bottom-up coarse-grained particle method, derived directly from the molecular dynamic formulation, and governed by the Newton’s law with calibrated coarse-grained force field. DPD and its variants have been widely applied to polymer science [42,45], biofluids [6,34], and hydrodynamics [4,49], to name but a few. Thermal fluctuations and their correlations at mesoscale are important and appear as a stochastic term in the governing equations. To fully resolve molecular details and capture the underlying dynamics at the mesoscopic level, a time step at the order of a picosecond is needed [47], which greatly limits the potential of mesoscopic models to simulate in vivo biological or physical processes with a larger characteristic time scale.

One promising strategy to overcome such bottleneck is by leveraging parallel computing. Following the philosophy of “divide and conquer”, parallel computing divides the whole spatial domain of the system into multiple subsystems, where each subsystem will be simulated simultaneously on distributed computing nodes. However, the computing efficiency may be limited when the number of distributing cores are too many to efficiently communicate in a message passing scheme [6]. As an alternative, a parallel-in-time strategy known as “parareal” originally proposed by Lions, Maday and Turinici [26] has been widely accepted as a parallel-in-time integration method. In the parareal method, a single long-time simulation is divided into multiple consecutive time-domains and distributed onto $n$ parallel simulations for representing the system dynamics at time $t_1, t_2, \ldots, t_n$. Essentially, parareal adopts an iterative prediction-correction strategy, where each subsystem at different time will communicate between its neighbors in time and make prediction with correction until convergence [15]. Following the idea of the parareal method, Blumers et al. [5] proposed a supervised parallel-in-time algorithm for stochastic dynamics (SPASD), which couples the finite element (FEM) solver as a supervisor with a DPD solver. Following the guidance of the FEM solver, the SPASD method converges (exponentially) fast after a number of iterations.

To illustrate how the multiscale parareal method can facilitate realistic simulations and reveal detailed dynamics, we consider an interesting biology model, the zebrafish, as our experimental model and simulate its hemodynamics in the hindbrain. Zebrafish has been receiving an increasing amount of attention in fields such as cancer biology, vascular angiogenesis and remodeling. Specifically, vascular remodeling is a biological process, where some unnecessary vessels in primature and unstructured vascular networks formed from angiogenesis [39] are subsequently pruned to improve the transport properties of the entire vasculature. As a result, an efficient matured network characterized by a hierarchical structure is achieved [28]. Although angiogenesis is considered to be mainly driven by chemical signals secreted from hypoxic tissues, recent evidence has shown that hemodynamics plays also an important role in vessel pruning [3,19]. Shear stress on vessel wall stimulates vessel regression through chemical signaling [14,38], which is followed by retraction, apoptosis, and reintegration of endothelial cells. Due to the difficulty of estimating the force acting on the vessel wall, however, the relationship between the hemodynamic parameters and the vessel pruning has not been fully clarified. Zebrafish possesses a unique feature that its embryo is transparent, so that in-vivo imaging of vessel structures and blood cell dynamics is possible. In the present study, we propose to apply the multiscale parareal algorithm on simulating the blood flow in zebrafish hindbrain, where such simulation is restricted by the expensive computational cost and efficiency. To the best of the authors’ knowledge, this is the first attempt to clarify the hemodynamic parameters through detailed three-dimensional (3D) simulation of the zebrafish hindbrain.

Using the continuum formulation, the computational cost for solving 3D Navier–Stokes(NS) equations could be extremely high since a fine mesh is needed to discretize such a complex 3D spatial domain. Instead, an one-dimensional

Table 1 Features of the coarse solver and fine solver in SPASD

| Type     | Coarse Solver (CS) | Fine Solver (FS) |
|----------|--------------------|------------------|
| Method   | Continuum model    | Particle model   |
| Dimensionality | 1D Blood Flow Model | Three-dimension  |
| Code     | Serial             | Parallel         |
| Hardware support | CPU                | GPU              |

Hardware support CPU GPU
(1D) blood flow model is adopted to supervise the expensive mesoscopic simulator as it predicts vessel blood flow at satisfactory accuracy with a low computational cost [13,40]. The complex vascular network in the zebrafish hindbrain is represented as a 1D pipe-line network where vessel branches merge and diverge at bifurcation points. Along a single branch, the velocity and area at grid points are computed to characterize the local hemodynamics. The 1D model can be derived from the full 3D Navier–Stokes equations under the assumptions of incompressibility, axial dominance in velocity, and the absence of turbulence [40]. Many successful biological and biomedical applications have proven the versatility and low computational cost of the 1D blood flow model in predicting blood flow and pressure in coronary arteries [30,51], circle of Willis [16], or abdominal arteries [36,41]. Therefore, we adopt the 1D blood flow model in our continuum-based solver in this study.

We apply the multiscale parareal algorithm for accelerating massive Lagrangian simulations of blood flow at mesoscale to zebrafish hindbrain, see Fig. 1. We will also demonstrate that further computational speedup can be achieved by combining the present strategy with a classical spatial decomposition method. The reminder of this paper is organized as follows: we first introduce SPASD for the application of blood flow in Sects. 2.1 and 2.2. We then explain in detail the coupling scheme for on-the-fly communications between our coarse and fine solvers (CS/FS) in Sect. 2.3, containing the algorithmic architecture that enables real-time data transfer in massive parallel simulations. In Sect. 3, we present quantitative results of mesoscopic simulations accelerated by our SPASD framework. Finally, Sect. 4 contains a brief summary and discussion.

2 Algorithm

2.1 Methods

The high-fidelity Lagrangian simulator of SPASD employs DPD, which is referred as the “fine solver” (FS, as shown in Fig. 1). DPD is a widely used particle model for mesoscopic simulation of complex fluids, where each DPD particle is represented explicitly by its position and velocity. This approach offers a bottom-up approach that resolves motion with Newton’s equation and provides a flexible representation of various types of fluid. In particular, Pivkin et al. [34] developed a mixture of DPD particles and coarse-grained mesh-based cells to simulate the motion of red blood cells with ambient flows in vessels. In the multiscale parareal algorithm, FS is complemented by a low-fidelity 1D model for simulating blood flow in arteries (referred as the “coarse solver”, CS). The 1D model provides a macroscopic yet less detailed description of blood flow by assuming the dominance of blood flow in the axial direction and no perfusion. The 1D solver for blood flows can be executed on a single CPU with a low cost, and these results will then be transferred to DPD models as a guidance in each temporal sub-domain. Table 1 shows a comparison between the two models. More details about the DPD and 1D models are provided in the “Appendix”.
In this section, we introduce the details of SPASD. Notice that it is flexible to choose any combination of a stochastic particle model and its continuum counterpart other than the DPD/1D models [5] for SPASD (Algorithm 1). Such flexibility is warranted by the deliberate choice of the flow variable \( Q^n_k \), which serves as the medium between meso- and macro-scales (Table 2). An appropriate flow variable must be easy-to-extract and accurately describe the state of the fluid system. In this work, we take the flowrate along the local axial direction as the flow variable \( Q^n_k \) to couple the CS and FS models for the reason that its evolution describes the state of blood flow over time; \( k \) denotes the number of iteration and \( n \) is the index of temporal sub-domain. For the particle model, the axial flowrate can be estimated from the product of averaged particles velocity on a cross-sectional plane and its cross-sectional area.

### Algorithm 1 SPASD

1. Initialization: Advance in CS: \( G^{n+1}_0 = G(Q^n_0) \) for \( 0 \leq n \leq N - 1 \).
2. Main loop
   
   for \( k = 1, \ldots, K \) do
   
   DO PARALLEL for \( k - 1 \leq n \leq N - 1 \).

   Correction:
   
   - Filter solution to remove noise: \( \mathcal{D}\{Q^n_k\} \)
   - Advance with CS: \( G^n_{k+1} = G(\mathcal{D}\{Q^n_k\}) \)
   - Map the macroscopic state to microscopic state: \( \mathcal{R}\{Q^n_k\} \)
   
   - Skip mapping if \( Q^n_k - Q^n_{k-1} < \Delta_{th} \)
   - Advance with FS: \( \mathcal{F}(\mathcal{R}\{Q^n_k\}) \).

   Project the microscopic state to macroscopic state: \( \mathcal{P}^{n+1}_k = \mathcal{P}\{\mathcal{R}\{Q^n_k\}\} \).

   Compute the correction: \( \delta^{n+1}_k = \mathcal{P}^{n+1}_k - G^{n+1}_k \).

   END PARALLEL

   for \( n = k - 1, \ldots, N - 1 \) do

   Prediction:

   - if \( n = k - 1 \) do
     - Update: \( Q^{n+1}_k = \mathcal{P}^{n+1}_k \)
   
   else do
     - Advance with CS in serial: \( G^{n+1}_{c,k} = G(\mathcal{D}\{Q^n_{k+1}\}) \).

   Refinement:

   - Combine the correction and the prediction terms: \( Q^{n+1}_{k+1} = G^{n+1}_{c,k} + \delta^{n+1}_k \).

   end if

end

Break if the solution converges.

---

### 2.2 Supervised parallel-in-time algorithm for stochastic dynamics (SPASD)

In this section, we introduce the details of SPASD. Notice that it is flexible to choose any combination of a stochastic particle model and its continuum counterpart other than the DPD/1D models [5] for SPASD (Algorithm 1). Such flexibility is warranted by the deliberate choice of the flow variable \( Q^n_k \), which serves as the medium between meso- and macro-scales (Table 2). An appropriate flow variable must be easy-to-extract and accurately describe the state of the fluid system. In this work, we take the flowrate along the local axial direction as the flow variable \( Q^n_k \) to couple the CS and FS models for the reason that its evolution describes the state of blood flow over time; \( k \) denotes the number of iteration and \( n \) is the index of temporal sub-domain. For the particle model, the axial flowrate can be estimated from the product of averaged particles velocity on a cross-sectional plane and its cross-sectional area.

#### Table 2 Notation for selected variables

| Symbol | Description |
|--------|-------------|
| \( \mathcal{R} \) | Mapping operator: from CS to FS |
| \( \mathcal{P} \) | Projection operator: from FS to CS |
| \( \mathcal{D} \) | Filtering operator: noise filtering |
| \( \mathcal{G} \) | Coarse propagator: advancing in CS |
| \( \mathcal{F} \) | Fine propagator: advancing in FS |
| \( K \) | Number of iterations |
| \( N \) | Number of time subdomains |

SPASD is summarized in Algorithm 1 and represented graphically in Fig. 2. CS advances independently in the initialization. Since the governing equations of the macroscopic model are very different from those of the microscopic model, the form and dimension of the solution representing the macroscopic state substantially varies from those representing the microscopic state. The solution from CS can be mapped to a microscopic state via a mapping operation (denoted by mapping operator \( \mathcal{R} \)) while the converse mapping from FS to CS can be obtained by the projection operator \( \mathcal{P} \). Typical methods such as weighted kernel sampling and interpolation are employed for the mapping and projection operations. In the current application, the flowrate is computed at centerline points, which are shared by both models because the 1D geometry is extracted from the 3D geometry. Therefore, the mapping and projection operations are not required here. Moreover, noise filtering \( \mathcal{D} \) is essential to the success of SPASD because the noise from the FS would accumulate with time, which will subsequently lead to a divergence of refined solution from the true solution if left unattended. In this work, noise filtering is embedded in the advancement of CS.

First, the volume flowrate is an integral quantity over the entire cross-section, which can flatten the fluctuations from the FS. Moreover, the spatial derivative on the flowrate is only first order in the governing equations of CS to benefit the noise smoothing. Coupled with fixed boundary values, the system can effectively eliminate noise over time. Then, FSs advance in parallel starting from the same initial conditions with CS to compute their discrepancies at ending temporal
Fig. 2 A graphical representation of SPASD. The flow variable $Q^n_k$ is computed via correction, prediction, and refinement in each iteration. After initialization, FS $P$ will advance the system in parallel within its temporal sub-domain, encompassing other operations, i.e., mapping $R$, projection $P$, and $D$. The correction terms $\delta$ are computed at the end of each temporal sub-domain to correct the prediction in a serial propagation $CS$ in prediction and refinement. $a$ and $b$ present the first iteration, $c$ shows the second iteration starting from correction points, which will be used as refinements in the following prediction step to correct $CS$ (see Fig. 2b and c). Notice that the iteration can be terminated early since we observed that the solution converges fast.

We customized SPASD for the application of simulating zebrafish blood flow in this work. The present Algorithm 1 is essentially identical to the original SPASD except one addition – skip mapping if $Q^n_k - Q^n_{k-1} < \Delta_{th}$, where $\Delta_{th}$ a user-defined threshold. This modification is intended to hedge against the non-uniform particle distribution due to the explicit modeling of red blood cells (RBCs). Since the flowrate is computed using particle density, non-uniform particle distribution makes the flowrate slightly inaccurate, as demonstrated in the Y-bifurcation benchmark example. When the difference between two consecutive iterations $Q^n_k - Q^n_{k-1}$ is less than 10%, we found that skipping the mapping $R(Q^n_k)$ avoids inaccurate re-initialization of temporal sub-domains.

2.3 Multiscale framework

Coupling mesh-based/mesh-free models across scales can be challenging, since issues may arise from the innate stochasticity of meso-scale and the quantification of flow variable from the mesh-free model. In this section, we will explain in detail the coupling scheme between the two coupling models, namely DPD and 1D. For clarity, we will outline the interplay between the models in Sect. 2.3.2. We describe the implemented software architecture that enables real-time data transfer in massive parallel simulations in the “Appendix”.

2.3.1 Velocity computation

It is straightforward to compute the velocity in the 1D model. Since the computational domain of the 1D model is essentially a pipeline system, where each pipe is the centerline extracted from the 3D domain, the velocity can be readily computed at each centerline point as shown in Sect. 5.2. However, it takes additional steps to estimate the velocity for the DPD model. First, the 3D computational domain for the particle system can be decomposed into a list of non-overlapping computational cells (referred as “cells” in the following context), where the neighboring particles are lumped into one cell. Each cell is represented by its center referred to as the cellular node, as illustrated in Fig. 3. By averaging the dynamical state of particles in a cell as a whole, the mean flowrate for each cell can be approximated as

$$Q^D \approx \sum_i v_i \cdot \hat{n} \cdot A = \sum_i \frac{v_i \cdot \hat{n}}{h \cdot \rho},$$

where $v_i$ is the velocity of particle $i$, $\hat{n}$ is the direction of the centerline passing through the cell, and $A$ is the area of the cross-section. We refer to the cell thickness $h$ as the length of a computational cell along its centerline. The mean axial velocity is computed by $\sum v_i \cdot \hat{n} / (\text{Vol} \cdot \rho)$ with $\text{Vol} = A \cdot h$ being the cell volume and $\rho$ being the number density of DPD particles. The mean flowrate is estimated by the product of the mean axial velocity and the area of cross-section $A$. Note that the velocity at bifurcation cells is not computed since they is undefined. Because of the correspondence of computational domains between the two models, we can establish connections, which enable multi-scale communications transported by the flow variable.

2.3.2 Coupling scheme

Outlined in Table 3, the coupling scheme starts with associating particles to their corresponding cells after a propagation in the DPD model. The flowrate $Q^D$ in DPD is then computed and pushed into the 1D model as the initial condition. Then, a series of 1D simulations using the fetched $Q^D$
Table 3 Outline of the coupling scheme between the 1D and DPD systems

| ID model | DPD model |
|----------|-----------|
| 1 Run the DPD simulation. | Down-scaling: Associate particles to their respective cells; compute flowrate \( Q_D \) for each cell by summing over contributions from enclosed particles |
| 2 | Push flowrate for each cell \( Q_D \) on cellular nodes |
| 3 Fetch \( Q_D \) on centerline points | Push \( Q_D \) on cellular nodes |
| 4 Update \( Q_N \) with corrections as described in SPASD | Fetch updated flowrate \( Q_N \) on cellular nodes |
| 5 Push \( Q_N \) on centerline points from the next timestamp | Up-scaling: Tune particle velocity to match the corresponding cellular flowrate \( Q_N \) using \( Q_D \). |
| Repeat from 1 | Repeat from 1 |

as initial condition will be performed to compute an updated flowrate \( Q_N \) with corrections in serial. Subsequently, the computed flowrate will be pushed back to the DPD model as the initial conditions, i.e., particle velocities, which is commonly referred to as “up-scaling” in the context of multiscale coupling”. In particular, a universal mapping cannot effectively match the flow rates due to the stochastic nature of the particle system. Hence, we propose a modified mapping scheme pre-conditioned on the current state as following:

\[
\begin{align*}
    v_{i}^{\text{New}} & \left\{ \begin{array}{ll}
        Q_N \frac{A}{A} \hat{n}, & \text{if } 1 - \alpha < F < 1 + \alpha \quad (2) \\
        v_i + (v_i \cdot \hat{n})(\frac{Q_N}{Q_D} - 1) \hat{n}, & \text{otherwise}, \quad (3)
    \end{array} \right.
\end{align*}
\]

where \( F \equiv (Q_N - Q_D)/Q_N \) and \( \alpha \) is a pre-determined parameter taking a value between 0 and 1 to regulate the method selection. We use \( \alpha = 0.5 \) for this work based on our empirical tests. \( A \) is the average area of a cell, which we approximate as the cross-sectional area at the cellular node. We note that the position of each RBC is not re-initialized at the beginning of each SPASD iteration: the position of each RBC is inherited from the previous step with an updated velocity (momentum) for each particle.

The idea of this pre-conditioning mapping is to filter out numerical artifacts that may lead to divergence in this scheme. When the current flowrate \( Q_D \) and the target flowrate \( Q_N \) are on the same order of magnitude and non-zero, the particle velocity is modified according to Equation (3). However, \( Q_N/Q_D \) becomes unreasonably large if \( Q_D \) is close to zero, where this scenario typically occurs in the first iteration when the particle system (or part of it) was under initialization. In this case, Equation (3) cannot accurately tune particle velocities. Instead, the particle velocity is modified according to Equ. (2) when \( 1 - \alpha < F < 1 + \alpha \).

For each combination of the signs of \( Q_N \) and \( Q_D \), we tabulate the preferred method given the range of \( F \) in Table 4.

Since the 1D and the DPD model have their respective coordinate systems, a directional mapping \( \hat{n} \) for translating...
vectorial quantities between the solvers is necessary to couple them together. The flow orientation \( \hat{n} \) for each branch needs to be pre-assigned in the 1D model before performing simulations. However, the pre-assigned orientations will not affect the actual flow direction in the coupling scheme; the velocity value can be negative if there exists a mismatch. We illustrate more details on implementation in the “Appendix”.

### 3 Results

#### 3.1 Benchmark problem – Y bifurcation

We first consider a simple fluid flow in this Y bifurcation channel with Newtonian fluid assumption. The DPD pair interaction parameters are chosen as: \( \alpha = 30 \), \( \gamma = 4.5 \), \( \sigma = 3 \), \( s = 2 \), and \( r_c = 1 \). As shown in Fig. 5, we impose

![Fig. 5 Central-plume of the 3-dimensional bifurcating channel. The channel is composed of one parent pipe and two identical daughter pipes. For DPD, Dirichlet-type boundary conditions are enforced in the boundary zones. We sample statistics of the flow, such as velocity and flowrate, in sample zones. The length scale is 10⁻⁵ m](image)

where \( \bar{Q}^{\text{ref}} \) and \( \bar{Q}^{\text{sim}} \) are the reference and simulated results, respectively.

Moreover, the flowrate from DPD is an ensemble average over 1000 consecutive frames in a bin with size 2×2×2. We set the ensemble time length long enough and take average from 40 independent runs to minimize the effect of stochasticity. The averaged results are then compared against those of the reference. Note that the reference solution is obtained by running DPD simulations without SPASD. Although the noise is still present post-averaging, it is reduced to a manageable magnitude so that the validity of SPASD can be meaningfully inspected.

#### 3.1.1 Simple fluid

We first consider a simple fluid flow in this Y bifurcation channel with Newtonian fluid assumption. The DPD pair interaction parameters are chosen as: \( \alpha = 30 \), \( \gamma = 4.5 \), \( \sigma = 3 \), \( s = 2 \), and \( r_c = 1 \). As shown in Fig. 5, we impose

\[
\frac{\| \bar{Q}^{\text{ref}} - \bar{Q}^{\text{sim}} \|_2}{\| \bar{Q}^{\text{ref}} \|_2} \equiv \frac{\sum_{i=1}^{N}(Q_{i}^{\text{ref}} - Q_{i}^{\text{sim}})^2}{(\sum_{i=1}^{N}(Q_{i}^{\text{ref}})^2)^{1/2}},
\]
a Dirichlet boundary condition at the inlet for flowrate with a parabolic profile defined as $u(r) = c(R^2 - r^2)$, where $r$ denotes the radius, and $c$ is the magnitude. In this example, $c_{\text{inflow}}$ and $c_{\text{outflow}}$ are 0.02 and 0.0152, respectively. For this simple fluid example, the flowrates at the boundaries are prescribed by adjusting the velocity of particles in the boundary zones to match the desired values. For 1D simulations, the inlet boundary condition is properly non-dimensionalized and imposed with density and dynamic viscosity of 1060 kg/m$^3$ and 4 cP.

In Fig. 5, the temporal domain is divided into four non-overlapping sub-domains each of which is associated with an independent DPD system running concurrently. That is, the systems are initialized at $t = \{0, T/4, T/2, 3T/4\}$, and ended at $t = \{T/4, T/2, 3T/4, T\}$, respectively. We plot the refined flowrate at iteration 3 from DPD simulations in Fig. 6 at the end of each temporal sub-domain. The horizontal axis denotes the axial coordinate of sampled points on each branch. Furthermore, we tabulate the $l^2$ relative errors in Table 5 and found that $\epsilon$ is less than 1% for all sub-domains. Additionally, the numerical convergence is shown by the errors are on the same order across iterations. All of these results indicate a good agreement between the reference results and that of SPASD.

Moreover, we compare the velocity profiles and shear rates from DPD in Fig. 7 for further verification. A shear rate are obtained by first fitting a velocity profile to a parabolic equation, and then calculating its analytic gradient. Again, we found the reference and SPASD solutions match well for all iterations. Importantly, we observe that the results and errors stay approximately the same regardless of the number of iterations, indicating that one iteration is sufficient for convergence.

### 3.1.2 Complex fluid: Blood flow

In this example, we test our proposed model on the same Y-bifurcating channel with complex fluids where the fluid is a mixture of single particles and discrete RBCs at a hematocrit of 60% (180 RBCs). Illustrated similarly in Fig. 5, the complex fluid flow is driven by prescribing a sinusoidal flowrate over time at boundaries. The maximum flowrate at the parent and each daughter boundary is approximately 63 and 31 cP, respectively. We impose Dirichlet boundary conditions with an exponential velocity profile $u(r) = ae^{br} + c$, where $a$ and $c$ are adjusted to match the magnitude of the desired flowrate. The DPD parameters are set as shown in Table 6. The boundary conditions for 1D simulations are imposed in a similar way with the simple fluid example. Noticing

| Iteration | quarter (%) | half (%) | three-quarters (%) | full (%) |
|-----------|-------------|----------|--------------------|---------|
| Iteration 2 | 0.58        | 0.37     | 0.40               | 0.75    |
| Iteration 3 | 0.55        | 0.36     | 0.44               | 1.01    |

Table 5 $l^2$ relative error of the flowrate defined in Eq. (4), for the simple fluid example.
Fig. 7  a Velocity profiles and  b shear rates from the DPD system at specific times marked in the bifurcation channel Fig. 4. We plot the results from three iterations to show convergence, and see that the flow from SPASD is very close to that from the reference simulation.

Table 6 DPD parameters of pair interaction in Y-bifurcation example

|            | α   | γ   | σ   | s   | r_e |
|------------|-----|-----|-----|-----|-----|
| Wall      | 60  | 45  | 2.916 | 2   | 1   |
| Solvent   | 60  | 45  | 2.916 | 2   | 1   |
| Solvent   | RBC | 4   | 45  | 2.916 | 2   | 1   |
| RBC       | RBC | 4   | 30  | 2.381 | 0.5 | 1.5 |

that the 1D model only serves as a low-fidelity guide where certain levels of simplifications can be tolerated, the parameter setting in this example is inherent from the simple fluid case (Newtonian fluid), where a laminar blood flow with with parabolic velocity profile is assumed to be valid. In particular, the parabolic parameter $\alpha = \frac{4}{3}$. Note that this $\alpha$ is different from that in equ. 2.

We compare the refined flowrate for complex fluids at different snapshots in Fig. 8. This plot corresponds to iteration 3. The horizontal axis denotes the axial coordinate of sampled points on each branch. The tabulated $l^2$ relative error $\epsilon$ is shown in Table 7. In general, the results from the reference and SPASD correspond well. However, due to the non-uniformity of particle distribution in the complex fluids, the flowrate and velocity profiles match those of the reference less precisely than that in the simple fluid case. We will discuss and expand on this topic further in Sect. 4. Furthermore, we present the velocity profiles and shear rates with the reference values against radius $r$ at different sites and times in 9a and b. Since the shear stress at the wall is of particular interest owing to its mechanobiological effects on signaling activation, we compute the shear stress $\tau$ by averaging over bins that decompose the 3D domain. Also, we calculated the mean and variance of shear stresses from 50 independent simulations and plotted the results in Fig. 9c. As $r$ increases, $\tau_{\text{reference}}$ matches $\tau_{\text{SPASD}}$ much better. It is noticeable that the shear stress close to the center has a larger variance caused by the fact that the concentrations of RBCs and solvent particles are higher towards the center.

3.2 Realistic vascularture – zebrafish hindbrain

In this section, we consider a zebrafish hindbrain as the biological model (Fig. 10) to test our method on a complex domain. To visualize the zebrafish brain vasculature, we use a transgenic (Tg) zebrafish line, Tg(kdrl:mCherry-CAAX)mcv8, that can label plasma membrane of endothelial cells (ECs) specifically by expressing mCherry-CAAX under the control of kdrl (flk1) promoter [44]. For 3D confocal imaging, the zebrafish embryos are mounted in 1% low-melting agarose poured on a 35-mm-diameter glass-base dish (Asahi Techno Glass). Confocal images are taken with a FluoView FV1000 confocal upright microscope system (Olympus) equipped with water-immersion XLMPlan FL N 20x/1.00 NA objective lens (Olympus) regulated with Fluoview ASW software (Olympus). In order to visualize the
behavior of red blood cells (RBCs) flowing inside the vessels, we use double-transgenic Tg(kdrl;GFP);Tg(gata1;DsRed) embryos at 55 hours post-fertilization (hpf) that can label ECs and RBC with GFP and DsRed, respectively. High-speed movie images of zebrafish blood flow are taken at 68 frames per second (fps) using light sheet microscopy. Light sheet imaging is performed with a Lightsheet Z.1 system (Carl Zeiss) equipped with a water immersion 20x detection objective lens (W Plan Apochromat, NA 1.0), dual sided 10x illumination objective lenses (LSMF, NA 0.2), a pco.edge scientific CMOS camera (PCO) and ZEN software. We track RBCs manually with ImageJ software (National Institute of Health) in order to measure the velocity of RBCs. 14-26 RBCs (from 3 embryos) are tracked so as to obtain the blood flow velocities at the boundaries of the current computational domain. Blood velocity and direction at some boundaries are measured in vivo over a period of time (see Fig. 11). These measurement data are used for the boundary conditions in the present simulations.

As shown in Fig. 10, the three primary vessels, running along the body, carry blood from the heart to the rest of the body. The secondary vessels, sitting on top the primary vessels, branch from the primaries and circulate blood circumferentially. Our image analysis reveals 95 distinctive branches with 57 bifurcating junctions in our region of interest. At 2 days post-fertilization (dpf) developmental stage, the vessel diameters range from 10 to 20 micrometers, and the region is merely 224x197x160 micrometers in size (see Fig. 11). The compactness of the region and the complexity of the vascular network make it challenging for the 1D solver to model it accurately, but it is an ideal benchmark for our method.

**Ethics Statement:** The experiments using zebrafish were approved by the animal committee of National Cerebral and Cardiovascular Center (No. 19009) and performed according to the guidance of the Institute with respect to ethical issues and scientific care. To avoid unnecessary stress to the fish, Tricaine (Sigma-Aldrich, St. Louis, MO) was used as an anesthetic prior to all experiments.

### 3.2.1 Computational Setup

Our reconstructed 3D particle system contains 17,244,362 particles in total – 5,989,445 fixed particles as walls, 9,477,881 solvent particles and 3,554 RBCs, each of which is made of 500 particles. The hematocrit of our particles system matches the true hematocrit of 2 dpf larva at 43% [22]. To form a closed system with periodic boundary conditions, the inlets and outlets along the spinal direction are connected to a reservoir. As the outlets push blood into the reservoirs, the inlets take the blood from the reservoirs to replenish. To keep the implementation simple, we impose a no-flux boundary condition on the other four sides (i.e., top, bottom, front and back). A DPD boundary method for arbitrarily complex geometries developed by Li et al. [25] is used to automatically handle the 3D zebrafish brain vasculature. Table 8 shows the values of DPD parameters. More details of choosing parameters value can be found in [8,10,11,31,50]. The centerline for 1D is extracted based on the reconstructed 3D geometry and is used as the 1D computational domain. Since the 1D model only provides a coarse approximation of the blood flow, we set 1D parameters to the same values as the complex fluid case from above. We impose the RCR boundary condition to all outlets with a fixed value, i.e., $R_{tot} = 10^8$, $C = 10^{-8}$. We prescribe the same Dirichlet boundary conditions for both models, namely pulsatile flows mimicking the realistic conditions as shown in Fig. 12. The zebrafish was at 2 dpf, and the heart rate was approximately 175 beats per minute [21]. We re-normalized the flowrate profiles for each outlet/inlet so that the mean and max value match the experimental measurements. We further assume that the pulsatile flow is in-phase at inlets and outlets, although in reality there may exist a small phase difference.

To avoid numerical overflow and boost computational accuracy, the 1D and DPD system use their respective reduced units, which can be mapped to physical units. For the particle solver, the length scale is chosen to be...
Table 7 $l^2$ relative error of flowrate for the blood flow example

| Iteration | Quarter (%) | Half (%) | Three-quarters (%) | Full (%) |
|-----------|-------------|----------|--------------------|----------|
| Iteration 1 | 0.93 | 5.34 | 3.77 | 2.35 |
| Iteration 2 | 2.60 | 2.18 | 3.14 | 2.61 |
| Iteration 3 | 2.01 | 1.99 | 4.25 | 1.58 |

Fig. 9 a Velocity profiles and b shear rates and c shear stress at specific times marked in Fig. 5 for the blood flow example. We plot these quantities and compare them to those of the reference simulations. Shear stress is measured from 50 independent runs, and the error bar denotes one standard deviation from the mean.

$[L^{DPD}] = 1 \times 10^{-6}$m. This means that one unit length in the DPD system is equivalent to $10^{-6}$ meters of physical length. By matching the viscosity of real blood to that of the DPD fluid, the time scale is calculated to be $[T^{DPD}] = 1.2225 \times 10^{-3}$ s. On the continuum solver side, the length and time scales are chosen so that the flowrate matches that of the real blood in zebrafish. In this example, we used $[L^{1D}] = 1 \times 10^{-2}$m and $[T^{1D}] = 1$s for stability and accuracy. The details on mapping to physical units can be found in “Appendix 5.3”.

We divide the temporal domain into four sub-domains, as illustrated in Fig. 13. For the first iteration, $FS\, 1, 2, 3$ and 4 propagates the systems concurrently, one for each sub-domain. If the computational results in sub-domain 1 is accurate at the end of first iteration, it is unnecessary to proceed to further iterations. If further iterations are needed, $FS\, 1$ moves down to sub-domain 2 and the same applies to $FS\, 2$ and 3. The convergence in the global domain is guaranteed within four iterations.

3.2.2 Validation

We first simulate a full heart period using the 1D and DPD models independently. A qualitative comparison of the velocity field is presented in Fig. 14. Overall, we can observe an agreement between the (a) DPD and (b) 1D model both with light color at the top left and top right sites (inlets and outlets), indicating a faster blood flow. To be more quantitative, the
Fig. 10 3D geometry reconstruction of zebrafish brain vasculature. **a** vascular network in the entire body **b** a stack of 2D confocal microscope images of brain vasculature **c** reconstructed 3D vascular network in the brain **d** hindbrain vasculature for DPD simulation **e** extraction of the center line of each microvessel for 1D analysis (blue dot: end point, red dot: branching point, white dot: vessel center)

Table 8 DPD parameters of pair interaction in zebrafish hindbrain example

|        | $\alpha$ | $\gamma$ | $\sigma$ | $s$ | $r_c$ |
|--------|----------|----------|----------|-----|-------|
| Wall & Any | 20       | 45       | 2.916    | 2   | 1.5   |
| Solvent & Solvent | 20       | 45       | 2.916    | 2   | 1.5   |
| Solvent & RBC    | 4        | 45       | 2.916    | 2   | 1.5   |
| RBC & RBC        | 4        | 30       | 2.381    | 0.5 | 1.5   |

Computed velocities are then compared at 12 representative sites shown in Fig. 15, which include sites in the upstream and downstream (Fig. 16). Figure 17 presents a comparison of computed velocity from 1D and DPD where the results from DPD are averaged over 3 repetitive simulations. The extreme values at each timestep are recorded and plotted in Fig. 17 as well. The results from DPD simulations are considered as the reference solution, which fluctuate mildly due to the non-uniformity in the fluid flow. In general, the flow in 1D model has very little phase difference: the systoles for all vessels are approximately synchronized with little phase shift and with the inflow boundaries (Fig. 12). However, in the reference solution, there exists a noticeable phase differences at those sites, which may be due to the assumptions (e.g., neglecting branching angles) and simplifications of blood composition in the 1D model.

We then simulate a full heart period with SPASD, and compare the values of averaged velocity from DPD simulations in the temporal sub-domains against the reference values at those sites. Since we divide the temporal domain into four sub-domains, SPASD solution must converge in less than four iterations to be beneficial. Therefore, we run three iterations and plot the results against the reference solutions.
Fig. 12 A standard pulsatile flowrate in a single period. We normalize the standard pulsatile flow in accordance with the mean and maximum values of blood velocity at the boundaries.

in Fig. 18. The results exhibit strong oscillations in velocity at interfaces of sub-domains in iteration 1, suggesting gaps between the two models. Subsequently, blood flow velocity in the network then moves back to the reference as it recovers from the inaccurate initial value. Notice that the speed of recovery is dependent on factors such as the amount of deviation from the reference and its magnitude. However, it is not a linear response by the fact that a small local perturbation can have cascading reactions across the network and cause a lengthy equilibrium recovery time in complex networks. Iteration 2 is supposed to be initialized with values calculated via SPASD. Because the new values are within 10% threshold.

Fig. 13 The temporal domain is divided into four sub-domains. For the first iteration, FS 1, 2, 3 and 4 propagates the systems concurrently, one for each sub-domain. At the end of the first iteration, the solution in sub-domain 1 is accurate and does not need to be iteratively corrected.

If a second iteration is needed, FS 1 moves down to sub-domain 2. The same applies to FS 2 and 3. The convergence in the global domain is guaranteed within four iterations.

Fig. 14 Qualitative comparison for the velocity field from the reference (left) and a standalone 1D simulation with the same boundary conditions. The reference is obtained from an expensive DPD simulation without parallel-in-time.
from that of last iteration, no re-initializations were carried out at those sites. Instead, DPD models simply continue to march in time for the corresponding sub-domains. Iteration 2 skips the first sub-domain since the solution is obtained via FS alone. It continues from where Iteration 1 stops in subdomain 1. The same goes for the other sub-domains. After only two iterations, the result is very close to the reference solution. To demonstrate convergence, we plot the reference solution and the results after two iterations in Fig. 19. We run a third iteration and plot the results on the same figure for demonstration purposes. The difference after two and three iterations is less pronounced, meaning that SPASD can be terminated after the second iteration.

Remark 1 Given a fixed-size problem, the parallel speedup using conventional spatial decomposition (SD) is limited by Amdahl’s law [1], meaning that we cannot gain any speedup by simply using more computational resources. This work aims to use the temporal decomposition (SPASD) in conjunction with SD to achieve extra speedup when more computational resources become available. We define the extra speedup factor of SPASD as

$$S = \frac{T_{SD}}{T_{SPASD}},$$

(5)

in which $T_{SPASD}$ is the wall time of SPASD, and $T_{SD}$ is the wall time of SD being serial in time. For a meaningful comparison, the number of GPUs to clock $T_{SD}$ is the same as that of each temporal subdomain in SPASD. The speedup factor can be written as

$$S = \frac{T_{serial}}{T_{SPASD}} = \frac{\sum_{i=1}^{N} \tau_{j}^{DPD}}{\sum_{j=1}^{K} \left( \tau_{j}^{DPD} + \tau_{j}^{1D} + \tau_{j}^{Op} \right)} \lesssim \frac{N}{K},$$

(6)

where $\tau_{j}^{DPD}$, $\tau_{j}^{1D}$, and $\tau_{j}^{Op}$ are the wall time taken by the DPD, 1D solver, and operations (noise-filtering, up-scaling, and down-scaling) at iteration $i$. $N$ is the number of temporal sub-domains and $K$ denotes the number of iterations to convergence. Ideally, $\tau_{j}^{DPD}$ keeps the same at different iteration, and the total wall time per iteration is disproportionately

Fig. 16 Final snapshot of the DPD reference simulation in the zebrafish brain vasculature. A snapshot of a RBCs, b wall particles (blue), and c fluid particles (red) in the vasculature. Outputs of the zebrafish vasculature are connected by a reservoir where we impose periodic boundary condition
Fig. 17  Quantitative comparison of predicted velocity from CS (labeled as ‘1D solver’) and FS (labeled as ‘Reference’) at the 12 sites. The range of averaged particle velocity is indicated by the shaded area from three repetitive SPASD simulations.

Fig. 18  Comparison of averaged velocity in SPASD at the 12 sites and the reference data. The resulting average speed at 12 sites marked in Fig. 15 are shown here. The reference data comes from three standalone FS simulations for comparison.
Results in Fig. 18 in the second and third iterations are re-plotted. The results are very close to the reference solutions after only two iterations. We run a third iteration for demonstration purposes.

### Table 9. Extra speedup factor for the zebrafish simulations

| Zebrafish | T\textsubscript{SPASD}(s) | Extra Speedup | Extra Speedup Bound | T\textsubscript{DPD}(s) | T\textsubscript{ID}(s) |
|-----------|----------------|---------------|---------------------|-----------------|---------------------|
| iter. 1   | 2190           | 3.79          | 4                   | 2076            | 114                 |
| iter. 2   | 2153           | 1.91          | 2                   | 2067            | 86                  |

We decompose the temporal domain into four subdomains where each subdomain is performed on 3 NVIDIA Tesla V100 GPUs. The wall time (WT) of SPASD and theoretical extra speedup bound is also provided with the measured speedup. The same simulation with only the DPD solver running in serial (T\textsubscript{serial}) would take 8301 seconds.

4 Summary & discussion

In this paper, we demonstrated how to accelerate massive Lagrangian simulations of blood flow in a zebrafish hindbrain by employing the parallel-in-time algorithm SPASD. The

Remark 2 However, it is not possible to compare quantitatively the final position of individual RBCs in DPD and SPASD. There are two reasons: First, the DPD system is stochastic, which indicates that the trajectory of the same RBC is different given the same initial and boundary condition but with randomized solvents. Secondly, it is impossible to trace RBCs since their momentum will be re-initialized in DPD simulations at the beginning of each SPASD iteration. However, the momentum and local density are conserved before and after re-initialization, preserving statistical quantity such as local flow rate, RBC concentration and momentum.
framework was first demonstrated on a simple Y-bifurcation geometry. Physical quantities, i.e., velocity, shear rate, shear stress and flowrate, were computed and compared against the reference solutions. Specifically, the $l^2$ relative error on flowrate is less than 1% for simple fluid and less than 3% for complex fluid on average. We then demonstrated our framework on a reconstructed zebrafish hindbrain, which is a complex vascular network with 95 branches and 57 bifurcations. Comparatively, SPASD is better suited for simulating of simple fluids than for complex fluids. As demonstrated in Y-bifurcation in Sect. 3.1, the flowrate and velocity profiles for a simple fluid match closer to those of the reference, which is largely due to the uniformity of particle distribution. In contrast, RBCs are modeled explicitly in the complex fluid. Particle non-uniformity is thus more pronounced because the local particle density is increased by the presence of the cells. In addition, the non-uniform scattering of cells causes non-uniform particle distribution at the network scale, which enlarges the discrepancy between flowrate from SPASD and the reference in Fig. 8. Despite this, the maximum normalized $l^2$ error is only on the order of one percent in our complex fluid example, and it does not grow over iterations. The above results demonstrate the efficacy and efficiency of our proposed multiscale coupling scheme SPASD in accelerating a Lagrangian simulation of blood flow informed with in-vivo imaging on a 2 dpf zebrafish, which provides a time-budgeting way to investigate mesoscopic biological processes with simulations.

When simulating a large system with limited sources, SD offers better efficiency than SPASD. This does not contradict the limitations of spatial decomposition listed in Sect. 1. Instead, this is drawn from our past benchmark on large-scale particle simulations, where we have demonstrated that SD scales very well with limited computational resources due to the absence of the CS and the need for a second iteration. However, when ample computational resources are available, the combination of SPASD and SD offers better efficiency than SD alone for long-time simulations. Moreover, we still need to further investigate the optimal way to balance spatial and temporal decomposition. That is, for a fixed system with limited resources, we should maximize the overall efficiency by allocating the number of spatial and temporal subdomains. Also, the overall efficiency depends on other factors, such as machine architecture, GPU/CPU models, the efficiency of the computer program, etc. A further systematic study on maximizing the efficiency of SPASD is worth pursuing. Admittedly, the 1D model is a low-fidelity description of this complex system whereas less iterations and a higher speedup can be achieved by replacing it by a more accurate description, which should include the non-newtonian effect.

For the zebrafish example, we note that the extra speedup is defined as the computational gain by employing temporal decomposition. Three GPUs per temporal domain is a good balance between available resources and computational efficiency. Admittedly, the zebrafish example serves only as a proof of concept to primarily show how SPASD can further accelerate a long-time mesoscopic simulation in a complex system together with conventional spatial decomposition. As shown in [5], the inter-nodal communication time will eventually nullify the benefits of additional cores for a large dynamical system with long-time integration. However, SPASD does not suffer from such issues, which makes it a more scalable algorithm for long-time simulations. A higher speedup can be achieved if more computational resources are available so that the long temporal domain can be divided into more subdomains. For example, if one is interested in studying the hemodynamics of zebrafish in 100 cycles, the extra speedup can grow linearly with the increase of temporal domains, achieving at the order of 50-fold speedup. Simulations of biological processes that undergo long-time evolution, such as oxygen/nutrients transports, thrombus formation/remodeling, or vascular angiogenesis, can benefit most from SPASD since the underlying biological processes could take minutes to days.

Moreover, we would like to clarify a misconception about SPASD – SPASD does not predict the path of individual cells. For example, it cannot predict the location of a particular RBC in a future temporal sub-domain given its current location. To expand the statement, SPASD cannot predict the distribution of RBCs in the vascular network across time sub-domains. Instead, SPASD is more suitable for simulating well-mixed fluids. In the case of complex fluids such as blood, we are assuming the mixture of plasma and RBCs are homogeneous in the network scope. The inability to track cells across temporal sub-domains is a limitation of SPASD. In fact, it is a limitation of parallel-in-time methods for dynamical systems in general. SPASD is not suitable to simulate processes that depend on the trajectories of individual particles.

There are many reasons that could lead to the discrepancy between the 1D and DPD model. First, it can be caused by the discrepancy of geometry between 1D and DPD. As shown in Fig. 20, the 1D model assumes axial symmetry while the reconstructed cross-section area may not has this symmetry. The effect on blood flow distribution can be amplified in the downstream, leading a more distinct difference. Second, 1D and DPD are based on different assumptions. Essentially, DPD is a mesoscopic stochastic Lagrangian solver, which can resolve detailed RBC motions, whereas 1D is a macroscopic model based on the assumption that velocity is predominantly along the vessel axis. Third, the discrepancy can be caused by different setting in boundary conditions. In DPD, we impose the velocity boundary condition at each inlet and outlet on both ends. For the vessels in that ends in the middle (such as Fig. 20 outlet 1), we assume that the vessel is closed at the end by setting the outlets as wall boundary condition.
Currently, we do not have experimental measurements inside the computational domain, so we cannot compare with the experiment for this case. However, this work focuses on coupling a mesoscopic stochastic Lagrangian solver with a macroscopic DG solver for blood flow simulation. We are currently preparing a systemic study with more detailed experimental data obtained with optical coherence tomography, which records the blood flow at a set of observation points. Furthermore, machine learning can be served as a surrogate model that enables fast assessments of the quantities of interest in many system [7,20]. There is a significant interest on coupling a conventional numerical solvers, e.g. finite volume or smoothed particle hydrodynamics, with a machine learning model as a surrogate for further enhancing the computational efficiency and accuracy [23,24,37,52]. We will address these issues in our future work.

**Conflict of interest** The authors declare that they have no conflict of interest.

## 5 Appendix

### 5.1 dissipative particle dynamics

Each DPD particle is represented explicitly by its position and velocity. Its motion is governed by Newton’s equations of motion [18]:

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \quad (7)$$

$$\frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i = \sum_{j \neq i} (\mathbf{F}^C_{ij} + \mathbf{F}^D_{ij} + \mathbf{F}^R_{ij}), \quad (8)$$

where \( t, \mathbf{r}_i, \mathbf{v}_i \) and \( \mathbf{F}_i \) denote time, position, velocity, and force, respectively. The force comprises three components: conservative force \( \mathbf{F}^C_{ij} \), dissipative force \( \mathbf{F}^D_{ij} \), and corresponding random force \( \mathbf{F}^R_{ij} \) from particle \( j \) within a radial cutoff \( r_c \) of particle \( i \). They are expressed as [17]:

$$\mathbf{F}^C_{ij} = \alpha_{ij} \omega_C(r_{ij}) \mathbf{e}_{ij}, \quad (9)$$

$$\mathbf{F}^D_{ij} = -\gamma_{ij} \omega_D(r_{ij}) (\mathbf{e}_{ij} \cdot \mathbf{v}_{ij}) \mathbf{e}_{ij}, \quad (10)$$

$$\mathbf{F}^R_{ij} = \sigma_{ij} \omega_R(r_{ij}) \xi_{ij} \Delta t_f^{-1/2} \mathbf{e}_{ij}, \quad (11)$$

where \( \mathbf{e}_{ij} = \mathbf{r}_{ij}/r_{ij} \) is the unit vector between particles \( i \) and \( j \), and \( \mathbf{v}_{ij} \) is the velocity difference. \( \Delta t_f \) is the time step of the fine propagator, and \( \xi \) is a symmetric Gaussian random variable with zero mean and unit variance [17]. The strengths of the conservative, dissipative, and random force are \( \alpha_{ij}, \gamma_{ij} \) and \( \sigma_{ij}, \) respectively. The interaction between pairs of particles are regulated by weighting functions \( \omega_C(r_{ij}), \omega_D(r_{ij}) \) and \( \omega_R(r_{ij}) \). Most importantly, the balance between dissipation and thermal fluctuation is maintained by the fluctuation-dissipation theorem [9]:

$$\sigma_{ij}^2 = 2\gamma_{ij}k_B T, \quad \omega_D(r_{ij}) = \omega_R^2(r_{ij}), \quad (12)$$

where \( k_B T \) is the Boltzmann energy unit. A common choice for the weight functions is \( \omega_C(r) = 1 - r/r_c \) and \( \omega_D(r) = \omega_R^2(r) = \left(1 - r/r_c\right)^2 \) for \( r \leq r_c \) and zero for \( r > r_c \). For simplicity, \( k_B T \) and the mass of a particle are taken as the energy unit and mass unit, and their values are set to one.

#### 5.1.1 RBC and solvent

The particle-based blood flow model is based on the DPD approach [10,34]. In this model, the blood is composed of...
solvents and red blood cells. The solvent is modeled with single-particles subject to only pairwise forces described in Sect. 5.1, and red blood cells are modeled discretely with bonded-particles. Membrane viscosity, elasticity and bending stiffness can be recovered with a spring-network that resembles a triangular mesh on a 2D surface. The shape of the viscoelastic membrane is maintained by a potential derived from a combination of bond, angle and dihedral interactions. The bonds between particles have an attractive and a repulsive component mimicking springs. The attractive potential adopts the form of the wormlike chain potential and is given as

\[ V^{WC}_L = \frac{k_B T h_m}{4 p} \left( 3 - \frac{h}{h_m} \right)^2, \]

where \( k_B T \) 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

\[ V^{POW}_P = \begin{cases} -k_p (m-1) h^{m-1}, & m \neq 1, \\ -k_p \log(h), & m = 1, \end{cases} \]

where \( m \) is a positive coefficient, and \( k_p \) is a force coefficient. Additionally a viscous component damping the springs is realized by a dissipative force, as well as the corresponding random force, given as

\[ F^{D}_{ij} = -\gamma^T v_{ij} - \gamma^C (v_{ij} \cdot e_{ij}) e_{ij}, \]

\[ F^{R}_{ij} = \sqrt{2 k_B T} \left( \sqrt{2 \gamma^T (dW^S_{ij} \cdot I/3) + \sqrt{3 \gamma^C - \gamma^T} tr[dW^S_{ij}]I} \right), \]

where \( \gamma^T \) and \( \gamma^C \) are dissipative coefficients, and \( v_{ij} \) is the relative velocity. \( dW^S_{ij} \) is the symmetric component of a random matrix of independent Wiener increments \( dW_{ij} \).

RBC’s structural stability is maintained by the area and volume constraints given by

\[ V_{area} = k_s (A^t - A_0^t)^2 + \sum_j k_j (A_{j} - A_{0,j})^2, \]

\[ V_{volume} = k_v (V^t - V_0^t)^2, \]

where \( j \) is the triangle index. \( k_s, k_j, \) and \( k_v \) are the area stiffness coefficient, local area, and global volume constraint coefficients, respectively. The instantaneous area and volume of a 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. Lastly, the bending resistance of the membrane is modeled by

\[ V_{bending} = \sum_j k_b \left( 1 - \cos(\theta_j - \theta_0) \right), \]

where \( k_b \) is the bending constant, and \( \theta_j \) is the angle between two neighboring triangles with the common edge \( j \). We refer interested readers to references [10,34] for more details.

### 5.2 1D blood flow model

Under the assumptions of axial symmetry and radial displacement of the vessel wall (Fig. 21a) [4,40], the modeling of blood flow in a compliant vessel can be reduced from full-order Navier–Stokes equations to an 1D model. The 1D model has been widely used for blood flow simulations at a moderate [40] and low [33] Reynolds number. In our implementation, it serves as the low-fidelity model supervising the high-fidelity particle model because of its balance between its speed and accuracy.

The 1D model builds on mass conservation and momentum equations [40]:

\[ \frac{\partial A}{\partial t} + \frac{\partial A U}{\partial x} = 0, \]

\[ \frac{\partial U}{\partial t} + \frac{1}{\rho} \frac{\partial U^2}{\partial x} + \frac{1}{\rho} \frac{\partial p}{\partial x} = -K_f U, \]

where \( x \) is the axial coordinate along the vessel, and \( A(x, t) \) and \( p(x, t) \) are the cross-sectional area and intra-luminal pressure, respectively. \( K_f \) is the friction loss coefficient representing the viscous resistance of flow per unit length along the vessel. It can be expressed as \( K_f = \frac{2 \alpha \mu}{\rho A^2} \), where \( \alpha = 4/3 \) assumes a laminar blood flow with parabolic velocity profile. We assume no tapering effect such as stenosis or atherosclerosis by setting \( S = 0 \).

The system is closed with Laplace’s law on pressure-area:

\[ p = p_{ext} + \beta (\sqrt{A} - \sqrt{A_0}), \]

\[ \beta = \frac{\sqrt{\pi} h E}{(1 - \nu^2) A_0}, \]

The pressure \( p \) is the summation of the external pressure \( p_{ext} \) and pressure variation, which is characterized by arterial wall compliance \( \beta \) and square root of area difference. The Poisson ratio \( \nu \) is taken to be 0.5, and \( h \) and \( A_0 \) are the vessel wall thickness and reference cross-sectional area. \( E \) is the Young’s modulus of the vessel wall.

We use the second-order Adams-Bashforth scheme to discretize and compute the time integration. For spatial dis-
cretization, the whole arterial network is decomposed into \( N \) non-overlapping domains. Each domain can be further subdivided into elements. Within each element, the solution is formulated as a linear combination of orthogonal Legendre polynomials. At element interfaces, the solver uses the discontinuous Galerkin method to render the discontinuous numerical solutions. We refer interested readers to [40] for details.

The imposed boundary condition must satisfy conservation of mass, which constrains the flow rate at outlets. For the DPD model, we can impose Dirichlet type boundary conditions at inlets and outlets. However, for the 1D system, this would lead to a defective boundary condition [12] and a problem with well-posedness [35]. An alternative to the defective boundary condition is the Windkessel model [48], which is analogous to open-loop circuits. This approach captures the effective resistance (R) and compliance (C) of the neglected downstream vessels. In particular, we apply a three element “Windkessel” RCR boundary conditions to each of the outlets, as shown in Fig. 21(b). The RCR boundary conditions are given as:

\[
q_{in} = \frac{P(A^*) - P_m}{R_p} = C \frac{dP_c}{dt} + \frac{P_m - P_{out}}{R_d}
\]  

where \( R_p, R_d, C, A^* \) and \( q_{in} \) represent proximal and distal resistance, capacitance, area and outflow at 1D outlets. We tune the values of each resistor and capacitor so that the outflow rate and pressure match those of experimental measurements. For more detail, we refer interested readers to [2].

**5.3 Mapping to physical units**

For the DPD model, we set the length scale \([L^{DPD}] = 1 \times 10^{-6} \text{m}\). This means that one unit length in the DPD system is equivalent to \(10^{-6}\) meters of physical length. By matching the viscosity of real blood to that of the DPD fluid, the time scale \([T^{DPD}]\) can be evaluated with relation:

\[
[T^{DPD}] = [L^{DPD}] \frac{\nu^P}{\nu^M} \frac{\mu^M}{\mu^P} (s),
\]  

where \( \mu \) is the RBC membrane shear modulus, and \( \nu \) is the viscosity. The superscripts \( M \) and \( P \) denote the model and physical units, respectively. For the blood simulation, we use the viscosity of plasma \( \nu^P = 1.2 \times 10^{-3} \text{ Pa s} \) and the RBC membrane shear modulus \( \mu^P = 4.5 \times 10^{-6} \text{ Nm}^{-1} \):

\[
[T^{DPD}] = 10^{-6} \frac{1.2 \times 10^{-3}}{21.8132} \frac{100}{4.5 \times 10^{-6}} = 1.2225 \times 10^{-3} \text{s}.
\]  

Here, we set \( \mu^M = 100 \) and measured \( \nu^M = 21.8132 \) from running benchmark simulations.

**5.3.1 Parallel implementation**

The continuum and particle solvers are, by themselves, stand-alone solvers with complex code structures. Implementing codes into the solvers for coupling, while complying with the structure of the software, is a challenging task. To minimize code refactoring, we use Multiscale Universal Interface (MUI). MUI is a library that facilitates the concurrent coupling of heterogeneous solvers [43]. It integrates MIMD (Multiple instruction streams, multiple data streams) and an asynchronous communication protocol to handle inter-solver information exchange, irrespective of intra-solver MPI implementation. Coupling via MUI does not pollute the MPI communicator space, and therefore requires less code refactoring.

We illustrate the information-exchange pipeline in Fig. 22a. In this step, CS sends the volume-flux at various times to the FSs as initial conditions. Each FS receives...
its initial condition and resets its state to match the condition. Because there is no temporal overlap among the fine-propagations, they can be carried out concurrently. For each FS, its simulation domain is decomposed into multiple sub-domains for spatial acceleration, which creates an additional layer of parallelism. The results are then sent back to the coarse-solver.

The dual level of parallelism (i.e. spatial and temporal) is achieved owing to the clever arrangement of intra-solver and inter-solver communicators. The inter-solver communication is managed by MUI through MPI inter-communicators, which establish links between the CS and each of the FSs, as illustrated in Fig. 22b. In contrast, inter-solver communication is enabled by MPI intra-communicators. The master of each propagator shares the incoming information among the associated worker processes, each of which handles a sub-domain. At the end of a fine-propagation, the master pools information from the worker processes. The gathered data are then sent back through the inter-solver communicator.

References

1. Al-hayanni MAN, Xia F, Rafiev A, Romanovsky A, Shafik R, Yakovlev A (2020) Amaldh’s law in the context of heterogeneous many-core systems-a survey. IET Comput Digital Tech 14(4):133–148
2. Alastruey J, Parker KH, Peiró J, Sherwin SJ (2008) Lumped parameter outflow models for 1-D blood flow simulations: effect on pulse waves and parameter estimation. Commun Comput Phys 4(2):317–336
3. Bernabeu MO, Jones ML, Nielsen JH, Kruger T, Nash RW, Groen D, Schmiechek S, Hetherrington J, Gerhardt H, Franco CA, Coveney PV (2014) Computer simulations reveal complex distribution of haemodynamic forces in a mouse retina model of angiogenesis. J R Soc Interface 11(99):20140543
4. Bian X, Li Z, Adams NA (2018) A note on hydrodynamics from dissipative particle dynamics. Appl Math Mech (Engl Ed) 39(1):63–82
5. Blumers AL, Li Z, Karniadakis GE (2019) Supervised parallel-in-time algorithm for long-time lagrangian simulations of stochastic dynamics: Application to hydrodynamics. J Comput Phys 393:214–228
6. Blumers AL, Tang YH, Li Z, Li X, Karniadakis GE (2017) Gpu-accelerated red blood cells simulations with transport dissipative particle dynamics. Comput Phys Commun 217:171–179
7. Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE (2021) Physics-informed neural networks (pinn) for fluid mechanics: A review. arXiv preprint arXiv:2105.09506
8. Chang HY, Li X, Karniadakis GE (2017) Modeling of biomechanics and biohomeology of red blood cells in type 2 diabetes mellitus. Biophys J 113(2):481–490
9. Espanol P, Warren P (1995) Statistical mechanics of dissipative particle dynamics. EPL (Europhysics Letters) 30(4):191
10. Fedosov DA, Caswell B, Karniadakis GE (2010) A multiscale red blood cell model with accurate mechanics, rheology, and dynamics. Biophys J 98(10):2215–2225
11. Fedosov DA, Pan W, Caswell B, Gompper G, Karniadakis GE (2011) Predicting human blood viscosity in silico. Proc Nat Acad Sci 108(29):11772–11777
12. Formaggia L, Gerbeau JF, Nobile F, Quarteroni A (2002) Numerical treatment of defective boundary conditions for the Navier-Stokes equations. SIAM J Numer Anal 40(1):376–401
13. Formaggia L, Lamponi D, Quarteroni A (2003) One-dimensional models for blood flow in arteries. J Eng Math 47(3–4):251–276
14. Franke RP, Gräfe M, Schnittler H, Seifige D, Mittermayer C, Drenckhahn D (1984) Induction of human vascular endothelial stress fibres by fluid shear stress. Nature 307(5952):648–649
15. Gander MJ, Vandewalle S (2007) Analysis of the parareal time-parallel time-integration method. SIAM J Sci Comput 29(2):556–578
16. Grinberg L, Cheever E, Anor T, Madsen JR, Karniadakis G (2011) Modeling blood flow circulation in intracranial arterial networks: a comparative 3d/1d simulation study. Ann Biomed Eng 39(1):297–309
17. Groth RD, Warren PB (1997) Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. J Chem Phys 107(11):4423–4435
18. Hoogerbrugge P, Koelman J (1992) Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. EPL (Europhysics Letters) 19(3):155
19. Huang C, Sheikh F, Holland M, Cai C, Becker D, Chu PH, Evans S, Chen J (2003) Embryonic atrial function is essential for mouse early development, cardiac morphogenesis and angiogenesis. Development 130(24):6111–6119
20. Karniadakis GE, Kevekidis IG, Lu L, Perdikaris P, Wang S, Yang L (2021) Physics-informed machine learning. Nat Rev Phys pp. 1–19
21. Kopp R, Schwerte T, Pelster B (2005) Cardiac performance in the zebrafish bloodstream mutant. J Exp Biol 208(11):2123–2134
22. Lee J, Chou TC, Kang D, Kang H, Chen J, Baek KL, Wang W, Ding Y, Carlo DD, Tai YC, Hsiao TK (2017) A rapid capillary-pressure driven micro-channel to demonstrate neutronian fluid behavior of zebrafish blood at high shear rates. Sci Rep 7:1980
23. Li A, Chen R, Farimani AB, Zhang YJ (2020) Reaction diffusion system prediction based on convolutional neural network. Sci Rep 10(1):1–9
24. Li A, Farimani AB, Zhang YJ (2021) Deep learning of material transport in complex neurite networks. Sci Rep 11(1):1–13
25. Li Z, Bian X, Tang YH, Karniadakis G (2018) A dissipative particle dynamics method for arbitrarily complex geometries. J Comput Phys 355:534–547
26. Lions JL, Maday Y, Turinici G (2001) Résolution d’EDP par un schéma en temps pararéel. Comptes Rendus l’Academie des Sci Phys 355:534–547
27. Liu Y, Liu WK (2006) Rheology of red blood cell aggregation by computer simulation. J Comput Phys 220(1):139–154
28. Mirzapou-shafiyi F, Kametani Y, Hikita T, Hasegawa Y, Nakayama M (2020) Numerical evaluation reveals the effect of branching morphology on vessel transport properties during angiogenesis. bioRxiv Oct 13, pp 1–28
29. Monaghan JJ (2012) Smoothed particle hydrodynamics and its diverse applications. Annu Rev Fluid Mech 44:323–346
30. Mynard J, Maday Y, Turinici G (2001) Résolution d’EDP par un schéma en temps pararéel. Comptes Rendus l’Academie des Sci Phys 355:534–547
31. Peng Z, Li X, Pivkin IV, Dao M, Karniadakis GE, Suresh S (2013) A dissipative particle dynamics method for arbitrarily complex geometries. J Comput Phys 355:534–547
32. Peters A, Melchionna S, Kaxiras E, Lätt J, Sircar J, Bernaschi M, Bertsch M, Bisco A, Mucci S (2010) Multiscale simulation of cardiovascular flows on the ibm bluegene/p: Full heart circulation system at red-blood cell resolution. In: SC’10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, pp 1–10
33. Pindera MZ, Ding H, Athavale MM, Chen Z (2009) Accuracy of 1D microvascular flow models in the limit of low Reynolds numbers. Microvasc Res 77(3):273–280
34. Pivkin IV, Karniadakis GE (2008) Accurate coarse-grained modeling of red blood cells. Phys Rev Lett 101(11):118105
35. Quarteroni A, Veneziani A (2003) Analysis of a geometrical multiscale model based on the coupling of ODE and PDE for blood flow simulations. Multiscale Model. Simul. 1(2):173–195
36. Raghu R, Vignon-Clementel IE, Figueroa CA, Taylor CA (2011) Comparative study of viscoelastic arterial wall models in nonlinear one-dimensional finite element simulations of blood flow. J Biomech Eng 133(8)
37. Ray D, Hesthaven JS (2018) An artificial neural network as a troubled-cell indicator. J Comput Phys 367:166–191
38. Resnick N, Gimbrone MA (1995) Hemodynamic forces are complex regulators of endothelial gene expression. FASEB J. 9(10):874–882
39. Risau W (1997) Mechanisms of angiogenesis. Nature 386(6626):671–674
40. Sherwin S, Franke V, Peiró J, Parker K (2003) One-dimensional modelling of a vascular network in space-time variables. J Eng Math 47(3–4):217–250
41. Sherwin SJ, Formaggia L, Peiro J, Franke V (2003) Computational modelling of 1d blood flow with variable mechanical properties and its application to the simulation of wave propagation in the human arterial system. Int J Numer Meth Fluids 43(6–7):673–700
42. Symeonidis V, Karniadakis GE, Caswell B (2005) Dissipative particle dynamics simulations of polymer chains: scaling laws and shearing response compared to dna experiments. Phys Rev Lett 95(7):076001
43. Tang YH, Kudo S, Bian X, Li Z, Karniadakis GE (2015) Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers. J Comput Phys 297:13–31
44. Wakayama Y, Fukuhara S, Ando K, Matsuda M, Mochizuki N (2015) Cdc42 mediates bmp-induced sprouting angiogenesis through fmn3-driven assembly of endothelial filopodia in zebrafish. Dev Cell 32(1):109–122
45. Wang S, Li Z, Pan W (2019) Implicit-solvent coarse-grained modeling for polymer solutions via Mori-Zwanzig formalism. Soft Matter 15(38):7567–7582
46. Wang S, Ye T, Li G, Zhang X, Shi H (2021) Margination and adhesion dynamics of tumor cells in a real microvascular network. PLoS Comput Biol 17(2):e1008746
47. Wang Y, Li Z, Xu J, Yang C, Karniadakis GE (2019) Concurrent coupling of atomistic simulation and mesoscopic hydrodynamics for flows over soft multi-functional surfaces. Soft Matter 15(8):1747–1757
48. Westerhof N, Lankhaar JW, Westerhof BE (2009) The arterial windkessel. Med Biol Eng comput 47(2):131–141
49. Xia Y, Blumers A, Li Z, Luo L, Tang YH, Kane J, Goral J, Huang H, Deo M, Andrew M (2020) A GPU-accelerated package for simulation of flow in nanoporous source rocks with many-body dissipative particle dynamics. Comput Phys Commun 247:106874
50. Yazdani A, Deng Y, Li H, Javadi E, Li Z, Jamali S, Lin C, Humphrey JD, Mantzoros CS, Em Karniadakis G (2021) Integrating blood cell mechanics, platelet adhesive dynamics and coagulation cascade for modelling thrombus formation in normal and diabetic blood. J R Soc Interface 18(175):20200834
51. Yin M, Yazdani A, Karniadakis GE (2019) One-dimensional modeling of fractional flow reserve in coronary artery disease: Uncertainty quantification and bayesian optimization. Comput Methods Appl Mech Eng 353:66–85
52. Yu J, Hesthaven JS (2019) Flowfield reconstruction method using artificial neural network. Aiaa J 57(2):482–498

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.