Spatial and Temporal Acceleration of Mesoscopic Blood-flow Simulations

by

Ansel Lou Blumers

B.Sc., North Carolina State University; Raleigh, NC, 2013

M.Sc., Brown University; Providence, RI, 2016

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in Department of Physics at Brown University

PROVIDENCE, RHODE ISLAND

February 2020

This dissertation by Ansel Lou Blumers is accepted in its present form
by Department of Physics as satisfying the
dissertation requirement for the degree of Doctor of Philosophy.

Date_____ _____

George Em Karniadakis (乔治·埃姆·卡尼亚达克斯), Ph.D., Advisor

Recommended to the Graduate Council

Date_____ _____

Robert Pelcovits, Ph.D., Reader

Date_____ _____

Derek Stein, Ph.D., Reader

Date_____ _____

Zhen Li, Ph.D., Reader

Date_____ _____

Yosuke Hasegawa, Ph.D., Reader

Approved by the Graduate Council

Date_____ _____

Andrew G. Campbell, Dean of the Graduate School

Vita

Ansel Blumers attended North Carolina State University from 2009 to 2013 and earned a Bachelors of Science in Physics and minored in Mathematics, graduating Magna Cum Laude. After college, Ansel attended Brown University from 2013 to 2020, earning a M.S. in Physics in 2015 and his Ph.D. in Physics in 2020. In Summer 2018, Ansel was an intern at Idaho National Lab, resulting in the 2019 publication *A parallel-efficient GPU package for multiphase flow in realistic nano-pore networks*. During his time at Brown, he published *GPU-accelerated red blood cell simulations with transport properties* in 2017 and *Supervised parallel-in-time algorithm for long-time Lagrangian simulations of stochastic dynamics* in 2019.

Preface and Acknowledgments

I would like to first acknowledge my academic sibling Dr. Yu-Hang Tang. Dr. Tang introduces me to the field of high-performance computing that led to my first publication and subsequently a presentation at GPU Technology Conference. Secondly, I would like to acknowledge Prof. Zhen Li who co-advised me through my PhD career. Lastly, I would like acknowledge my Ph.D. advisor Prof. George Em. Karniadakis for his generous funding and support for the last five years.

Contents

List of Tables

List of Figures

Chapter One

Motivation: Need for Speed

1.1 Overview

Inherent stochastic fluctuations are important in many cellular and subcellular processes in biological systems, as well as in physical systems (soft matter, plasma dynamics, etc.) with timescales across many orders of magnitude. For such systems, a macroscopic deterministic description built on the continuum hypothesis is invalid [23], and therefore Lagrangian methods based on detailed atomic and molecular models are required [95]. However, because the atomistic details are explicitly modeled, the maximum time step is limited by the smallest oscillation period of the fastest atomic motion, which are typically of the order of several femtoseconds [30]. Consequently, the physical time scales accessible to those simulations are too short to address interesting long-time dynamics. This huge disparity in temporal scales, which can also happen at larger values of the smallest timescales, has not been addressed adequately in the computational literature on multiscale physical and biological systems.

Consider, for example, the problem of thrombus formation caused by platelet margination and adhesion to the arterial wall, where the characteristic time for platelet activation is a few microseconds. Lagrangian platelet models can only simulate the platelet dynamics up to a few minutes [98, 130], which is far below the time scale for those *in vivo* processes that usually require at least several hours or even days [56, 110]. With a time step of the order of 10^{-6} s, a clinically meaningful result for at least one-day time integration may require about 10^{11} time steps with explicit platelet models [32], which is computationally prohibitive.

Another example is the simulation of protein folding dynamics [38]. In an *ab initio* molecular dynamics (MD) simulation, the protein molecule and surrounding solvents

are treated as classical particles interacting through empirical energy functions. The MD trajectories provide extremely high spatial and temporal resolutions and can identify key intermediates of the folding processes. However, its computational cost hampers the study of large proteins due to their large number of degrees of freedom, greatly increasing the simulation time required to observe a single folding event, which is on the order of tens of microseconds to milliseconds [102]. For timescale on the order of milliseconds, MD simulations are challenging even for the specially designed ASICS machine - the longest continuous-trajectory atomic-scale simulation is a 2.936 millisecond simulation of NTL9 at 355 K [81]. Major conformational transitions triggered by enzymatic reactions occur on the millisecond to second time scale or longer [21]. Without algorithmic improvements, long-time integration of protein folding via *ab initio* MD simulations is a formidable problem.

The complexity due to the huge number of spatio-temporal degrees of freedom needed to resolve atomic or molecular details requires large amounts of computational resources and simulation time to capture the underlying kinetics and dynamics. A typical strategy to leverage parallel computing is dividing the spatial domain into many smaller subdomains so that all subdomains can be processed concurrently by many computing cores. We will refer to this as "conventional spatial decomposition" approach hereafter. Although it is relatively easy-to-implement and very effective, the strong scaling eventually plateaus as the number of unknowns per core becomes too small [10]. The lack of an efficient and stable numerical method remains a serious bottleneck for long-time simulations. Time parallel integration methods can break the bottleneck by decomposing the time domain and solving these subdomains parallel-in-time. Since the inception of time parallel integration methods over 50 years ago [92], many have improved and proposed new parallel-in-time methods. Parareal, proposed by Lions et al. [83], is a popular parallel-in-time scheme and has

demonstrated its versatility in various applications.

To resolve this bottleneck, I proposed a **S**upervised **P**arallel-in-time **A**lgorithm for **S**tochastic **D**ynamics (`SPASD`), which aims to significantly accelerate stochastic Lagrangian solvers for long-time simulations. `SPASD` is inspired by bottom-up coarse-graining projections that yield mean-field hydrodynamic behavior in the continuum limit. Inspired by Parareal, `SPASD` is built on the coupling of macroscopic and microscopic systems. The low-dimensional macroscopic system serves as a predictor to supervise the high-dimensional Lagrangian simulator, in a predictor-corrector type algorithm. The results of the Lagrangian simulation then correct the mean-field prediction and provide the proper microscopic details (e.g., consistent fluctuations, correlations, etc.). When combined with conventional spatial decomposition, `SPASD` can provide further acceleration.

Algorithmic improvements such as spatial and temporal decompsitions are only part of the answer. The effective use of hardware that run those simulations can provide additional acceleration. 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 [40, 14, 117, 80, 2, 101, 54, 103]. Most all modern super-computers are equipped with them as a way to boost computational power. In order to run massive simulations, I co-developed a GPU compatible simulator as a part of my PhD work.

1.2 Outline

Because we are studying the methodology and algorithm related to a Lagrangian particle solver, I will first give an overview of the statistical mechanics behind the solver Dissipative Particle Dynamics (DPD) in Chapter 2. The porting of the particle model to GPUs is described in detail in Chapter 3. There, I will introduce some algorithmic innovations that maximize scalability across multiple GPU nodes. The proposed parallel-in-time method for Lagrangian solvers, `SPASD`, is introduced in Chapter 4. Quantitative results including accuracy, convergence rate and parallel efficiency can also be found in that chapter. Lastly, in Chapter 5, I will demonstrate `SPASD` on blood flow simulations in realistic geometry of zebrafish hindbrain. For the demonstration, the simulation is spatially and temporally decomposed by the conventional spatial decomposition method and `SPASD`, respectively. On top of that, the simulation is run on GPUs for maximum acceleration.

Chapter Two

Theoretical Foundations of Dissipative Particle Dynamics

Dissipative particle dynamics (DPD) is a particle-based Lagrangian method for simulating simple and complex fluids at mesoscopic length. Its invention was prompted by the need to investigate mesoscopic structures of molecules or their collective behavior, without explicitly resolve the underlying atomistic details [131]. By directly working on the scale of interest, the atomistic dynamics is simplified by eliminating fast degrees of freedom while preserving the behavior of slow entities. It thus can capture the correct dynamics of complex fluids on larger spatial and temporal scales beyond the capability of conventional atomistic simulations [86].

Since the governing equations of DPD can be rigorously derived from microscopic and macroscopic dynamics, as I will dive into some details in this chapter, DPD possesses a solid theoretical foundation. Because of it, it has become a popular methodology with applications in complex fluids [96], polymer physic [52], computational biology [65] and more.

In this chapter, I will first consider its theoretical formulation in Section 2.1. After a short overview of the statistical mechanics behind DPD, I will then revisit the details of derivations of its governing equations with both top-down and bottom-up approaches in Section 2.2. The DPD parameters and their connections to macroscopic parameters will also be covered. Given the thesis focus mainly on hydrodynamics, I will only give a brief overview on the fundamental basis for coarse-graining underlying system of molecular dynamics, in relation to DPD, in Section 2.3. Finally, I will review one of many extensions of DPD - transport dissipative particle dynamics (tDPD), which grants the classic DPD the ability to model diffusion-reaction processes.

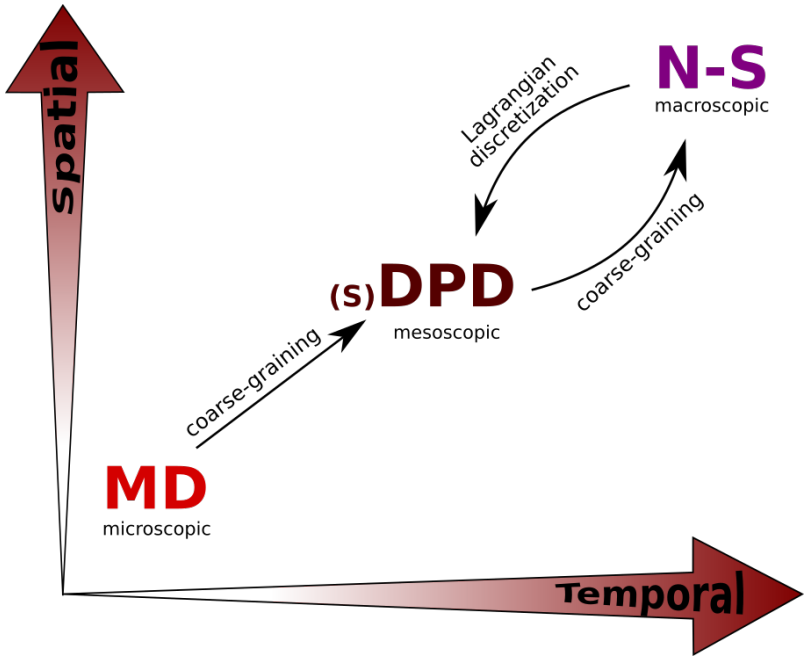


Figure 2.1: DPD's relation to microscopic and macroscopic methodologies, namely, molecular dynamics and Navier-stokes equations. Even though SDPD is a Lagrangian discretization of the Landau-Lifshitz-Navier-Stokes equation, there are strong similarities between the force equations of classic DPD and that of SDPD, as explain in Chapter 2.2.

2.1 Formulation of Dissipative Particle Dynamics (DPD)

Dissipative particle dynamics (DPD) was introduced as a way to study hydrodynamic phenomena at the mesocale by Kolemann and Hoggerbrugee [49, 59]. Español and Warren further developed the method [27] and proved that the equilibrium invariant distribution of a DPD system is the canonical ensemble of Gibbs distribution, which placed DPD on a firm physical ground. By virtue of theories of coarse-graining and Lagrangian discretization, the correspondence between DPD and microscopic and macroscopic methodologies were established, as shown in Fig. 2.1. These connections to well established methodologies solidifies its position as a scientifically sounding approach.

Following the approach taken by Español and Warren in [27], the equations of motion for DPD particles are formulated as continuous stochastic differential equations

$$\begin{aligned} \dot{\mathbf{r}}_i &= \dot{\mathbf{p}}_i/m, \\ \dot{\mathbf{p}}_i &= \mathbf{F}_i = \sum_{j\neq i} \mathbf{F}_{ij} = \sum_{j\neq i} \left(\mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R \right), \end{aligned} \tag{2.1}$$

where $\mathbf{r}_i$ and $\mathbf{p}_i$ are position and momentum of particle i. The conservative, dissipative and random forces are expressed as [27]

$$\begin{aligned} \mathbf{F}_{ij}^C &= a w_C(r_{ij}) \mathbf{e}_{ij}, \\ \mathbf{F}_{ij}^D &= -\gamma w_D(r_{ij}) (\mathbf{e}_{ij} \cdot \mathbf{v}_{ij}) \mathbf{e}_{ij}, \\ \mathbf{F}_{ij}^R &= \sigma w_R(r_{ij}) \xi_{ij} \mathbf{e}_{ij}, \end{aligned} \tag{2.2}$$

in which a, γ and σ are strengths, respectively. A common choice for the weighting function is

$$\begin{aligned} \omega_C(r) &= 1 - r/r_c, \\ \omega_D(r) &= \omega_R^2(r) = (1 - r/r_c)^s. \end{aligned} \tag{2.3}$$

for $r \leq r_c$ and zero for $r > r_c$. Because the forces depend only on relative position $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and relative velocity $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, DPD satisfies Galilean invariance and preserves angular momentum. In addition, to preserve linear momentum, each force is antisymmetric $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$. ξ_{ij} is a Gaussian white noise satisfying

$$\begin{aligned} < \xi_{ij}(t) > &= 0, \\ < \xi_{ij}(t) \xi_{kl}(t') > &= \left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk} \right) \delta(t - t'), \end{aligned} \tag{2.4}$$

where δ_{ij} and $\delta(t - t')$ are the Kronecker and Dirac delta, respectively. Inserting Eq.

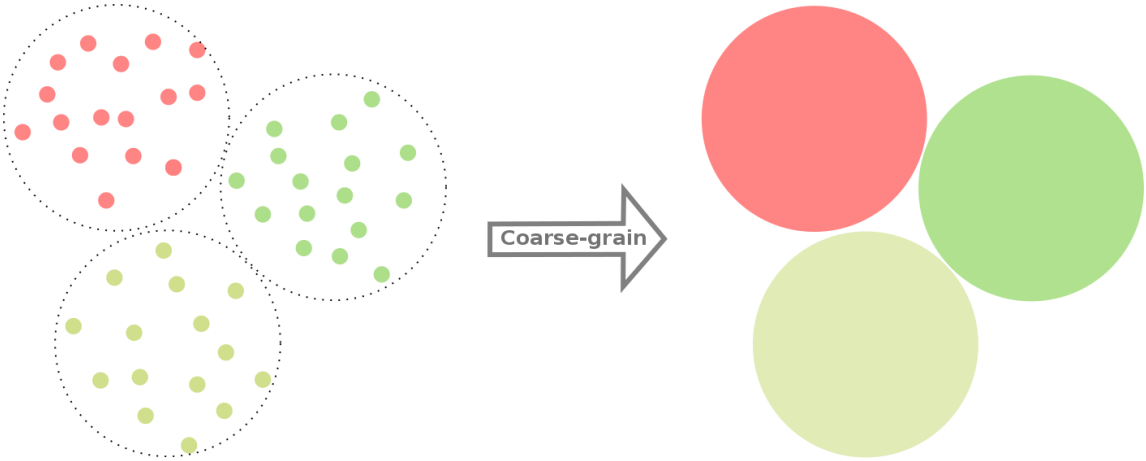


Figure 2.2: A single coarse-grain particle in DPD represents an entire cluster of molecules in MD. Similar to MD, the dynamics of DPD particles are computed from the time integration of Newton's equations of motion.

(2.2) into Eq. (2.1), we have the Langevin equations

$$\begin{aligned} d\mathbf{r}_i &= \frac{\mathbf{p}_i}{m_i} dt, \\ d\mathbf{p}_i &= \left(\sum_{j \neq i} \mathbf{F}_{ij}^C + \sum_{j \neq i} \mathbf{F}_{ij}^D \right) dt + \sum_{j \neq i} \sigma w^R(r_{ij}) \mathbf{e}_{ij} dW_{ij}, \end{aligned} \tag{2.5}$$

where dW_{ij} are independent increments of the Wiener process, i.e. $dW_{ij} dW_{kl} = \left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) dt$.

The conservation law $\frac{\partial f(\mathbf{x},t)}{\partial t} + \nabla \cdot \left[\dot{\mathbf{x}} f(\mathbf{x},t)\right] = 0$, with DPD forces, gives raise to the Fokker-Planck equation

$$\begin{aligned} \frac{\partial f(\mathbf{x},t)}{\partial t} &= \mathcal{L} f(\mathbf{x},t) = \mathcal{L}_C f(\mathbf{x},t) + \mathcal{L}_D f(\mathbf{x},t) + \mathcal{L}_R f(\mathbf{x},t), \\ \mathcal{L}_C &\equiv - \left(\sum_i \frac{\mathbf{p}_i}{m} \cdot \frac{\partial}{\partial \mathbf{r}_i} + \sum_{i,j}{}' \mathbf{F}_{ij}^C \cdot \frac{\partial}{\partial \mathbf{p}_i} \right), \\ \mathcal{L}_D &\equiv \sum_{i,j}{}' \mathbf{e}_{ij} \cdot \frac{\partial}{\partial \mathbf{p}_i} \left[\gamma w_D(r_{ij}) (\mathbf{e}_{ij} \cdot \mathbf{v}_{ij}) \right], \\ \mathcal{L}_R &\equiv \sum_{i,j}{}' \mathbf{e}_{ij} \cdot \frac{\partial}{\partial \mathbf{p}_i} \left[\frac{\sigma^2}{2} w_R^2(r_{ij}) \mathbf{e}_{ij} \cdot \left(\frac{\partial}{\partial \mathbf{p}_i} - \frac{\partial}{\partial \mathbf{p}_j} \right) \right], \end{aligned} \tag{2.6}$$

where $\mathcal{L}$ denotes the Fokker-Planck operator. In the context of particle methods, $f(\mathbf{x},t)$ is the *probability density function* (PDF) in phase space, and $\mathbf{x} = (r_1, r_2, ..., r_{3N}, p_1, p_2, ..., p_{3N})$ is the $6N$ dimensional coordinate.

The steady state solution to Eq. (2.6) is expected to be the Gibbs canonical ensemble, $f^{eq}(\mathbf{x}) = \frac{1}{Z}\exp\left[-(k_BT)^{-1}H(\mathbf{x})\right]$. Given the canonical ensemble is the equilibrium solution for a conservative system, we can obtain $\mathcal{L}_C f^{eq} = 0$ trivially. This forces $\mathcal{L}_D f^{eq} + \mathcal{L}_R f^{eq} = 0$, which can be satisfied by postulating

$$\begin{aligned} w_D(r) &= [w_R(r)]^2, \\ \sigma^2 &= 2\gamma k_B T. \end{aligned} \tag{2.7}$$

This constraint is the fluctuation-dissipative theorem for DPD, derived by Español and Warren in [27].

2.2 Correspondence to macroscopic hydrodynamics

As demonstrated in Sec. 2.1, a DPD system respects Galilean invariance and satisfies conservations of mass and momentum. On account of these properties, the macroscopic equivalence of DPD is the continuum Naviers-Stokes system. The correspondence between those systems has been established by Español [25] using Mori projection operator. Because the proof is lengthy and cumbersome, I will not revisit the derivations here. Instead, I will list the key equations and direct interested readers to Refs. [41, 89, 107] for technical detail.

The idea behind projection rests on the fact that, given a set of N variables and we are only interested in a subset of them or a set of a set of functions of the N

variables. Applying projection operator can leave us with fewer but relevant variables describing the dynamics. However, the reduced description of the system will have a memory and a random component that includes the eliminated variables. In short, projection techniques allows us to only keep relevant variables in a level that we are interested in.

As a consequence of eliminating variables, the evolution of coarse-grained variables cannot be predicated with certainty. The appropriate description of the coarse-grained system is thus in terms of a probability distribution for the relevant variables, and the evolution equation is the Fokker-Planck equation shown in Eq. (2.6).

The evolution of the relevant variables is described by equations of motion. For the microscopic description of classical mechanics, every degree of freedom in the system is included in the Hamiltonian equations. For hydrodynamics, the relevant variables are the mass, momentum and energy density fields. In thermodynamics, the relevant variables are the dynamical invariants of the system.

In the hydrodyanmics level of description, one of the relevant variables is the momentum density field $\mathbf{g}(\mathbf{r},t)$. Following through the Fokker-Planck equation, the time-evolution of the momentum can be described as

$$\frac{\partial \mathbf{g}(\mathbf{r},t)}{\partial t} = -c_s^2 \nabla \delta\rho(\mathbf{r},t) + \eta \nabla^2 \mathbf{v}(\mathbf{r},t) + \left(\zeta - \frac{2\eta}{3}\right) \nabla\nabla \cdot \mathbf{v}(\mathbf{r},t), \tag{2.8}$$

where $\rho(\mathbf{r},t)$ is the deviation from equilibrium mass density field [25]. c is the isothermal sound speed and determined only by $\mathbf{F}_{ij}^C$. The shear viscosity $\eta = \eta^C + \eta^D$ and bulk viscosity $\zeta = \zeta^C + \zeta^D$ are composed with contributions of conservative and

dissipative forces:

$$\begin{aligned}
\eta^C &= \beta \int_0^\infty \frac{1}{V} \left(\sum_{\mu\nu}^{C} (\tau), \mathcal{Q} \sum_{\mu\nu}^{C} \right) d\tau, \\
\left(\zeta^C - \frac{2}{3}\eta^C \right) &= \beta \int_0^\infty \frac{1}{V} \left(\sum_{\mu\mu}^{C} (\tau), \mathcal{Q} \sum_{\mu\mu}^{C} \right) d\tau, \\
\eta^D &= \beta \int_0^\infty \frac{1}{V} \left(\sum_{\mu\nu}^{D} (\tau), \mathcal{Q} \sum_{\mu\nu}^{D} \right) d\tau, \\
\left(\zeta^D - \frac{2}{3}\eta^D \right) &= \beta \int_0^\infty \frac{1}{V} \left(\sum_{\mu\mu}^{D} (\tau), \mathcal{Q} \sum_{\mu\mu}^{D} \right) d\tau,
\end{aligned} \tag{2.9}$$

with the following shorthand notations:

$$\begin{aligned}
\sum^{C} &\triangleq \sum_i \frac{\mathbf{p}_i}{m} \mathbf{p}_i + \sum_{i,j} \left(\mathbf{r}_i - \mathbf{r}_j \right) \mathbf{F}_{ij}^{C}, \\
\sum^{D} &\triangleq \sum_{i,j} \left(\mathbf{r}_i - \mathbf{r}_j \right) \mathbf{F}_{ij}^{D}.
\end{aligned} \tag{2.10}$$

Additionally, the following shorthand notation for a scalar product between two functions of phase space coordinates are also used in Eq. (2.9):

$$(\phi, \psi) \equiv \int f^{eq}(\mathbf{x}) \phi(\mathbf{x}) \psi(\mathbf{x}) d\mathbf{x} \equiv tr \left[f^{eq} \phi \psi \right]. \tag{2.11}$$

The set of equations above is known as the Green-Kubo relations. The dynamic and kinematic viscosities are given in terms of the stress tensor, which can be calculated from the Virial theorem via DPD forces $\mathbf{F}_{ij}^{C}$ and $\mathbf{F}_{ij}^{D}$. In other words, the Green-Kubo relations relate macroscopic transport coefficients to particle methods. Although Español demonstrated DPD satisfies the hydrodynamic equations at large spatial-temporal scale, it is not possible to quantify *a priori* the sound speed and

viscosities of a DPD system from force parameters in Eq. (2.2).

To avoid the difficulty of specifying those quantities *a priori*, we can discretize Lagrangian Naiver-Stokes [26]. The discretization approach was borrowed from smooth particle hydrodynamics (SPH) [88]. The essence of SPH is the re-formulation of an arbitrary function, denoted by A, with a kernel approximation:

$$A(\mathbf{r}) = \lim_{h\to 0} \int A(\mathbf{r}')W(\mathbf{r}'-\mathbf{r},h)d\mathbf{r}', \quad \int W(\mathbf{r}'-\mathbf{r},h)d\mathbf{r} = 1, \tag{2.12}$$

where the kernel $W(\mathbf{r}'-\mathbf{r},h)$ is also known as the normalized smoothing function, with h has the smoothing length. The gradient of the smoothing function can be defined as

$$\nabla W(\mathbf{r}'-\mathbf{r},h) = -(\mathbf{r}'-\mathbf{r})G(|\mathbf{r}'-\mathbf{r}|,h), \tag{2.13}$$

with $G \geq 0$.

Another important concept of SPH is approximating the integral with particle summation:

$$\int W(\mathbf{r}'-\mathbf{r},h)d\mathbf{r} <==> \sum_j W(|\mathbf{r}_i-\mathbf{r}_j|). \tag{2.14}$$

Because the integral of the smoothing function is unity, by summing over particle j that is within distance h from i, we get a number density $d_i = \sum_j W(|\mathbf{r}_i-\mathbf{r}_j|)$. With some trivial calculation, the Navier-Stokes in Lagrangian form

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{v}, \tag{2.15a}$$

$$\rho\frac{d\mathbf{v}}{dt} = -\nabla P + \eta\nabla^2\mathbf{v} + \left(\zeta+\frac{\eta}{3}\right)\nabla\nabla\cdot\mathbf{v}, \tag{2.15b}$$

can be discretized as

$$\dot{\rho}_i = -\rho_i \left(\nabla \cdot \mathbf{v}\right)_i, \tag{2.16a}$$

$$m_i \dot{\mathbf{v}}_i = -\frac{(\nabla P)_i}{d_i} + \frac{\eta(\nabla^2 \mathbf{v})_i}{d_i} + \frac{\eta \left(\nabla \nabla \cdot \mathbf{v}\right)_i}{3 d_i}. \tag{2.16b}$$

Using the 1^{st} and 2^{nd} laws of thermodynamics as constraints, the forces between particles can be represented by the transport coefficients and the derivative of the smoothing kernel:

$$\mathbf{F}_{ij}^C = \left(\frac{P_i}{d_i^2} + \frac{P_j}{d_j^2}\right) G_{ij} \mathbf{r}_{ij}, \tag{2.17}$$

$$\mathbf{F}_{ij}^D = -\left(\frac{5\eta}{3} - \zeta\right) \frac{G_{ij}}{d_i d_j} \mathbf{v}_{ij} - 5\left(\zeta + \frac{\eta}{3}\right) \frac{G_{ij}}{d_i d_j} \left(\mathbf{e}_{ij} \cdot \mathbf{v}_{ij} \mathbf{e}_{ij}\right), \tag{2.18}$$

$$\mathbf{F}_{ij}^R = \sum_j \left(A_{ij} d\overline{\overline{\mathbf{W}}}_{ij} + \frac{B_{ij}}{3} tr[d\mathbf{W}_{ij}]\right) \cdot \mathbf{e}_{ij}, \tag{2.19}$$

with the noise coefficients given by

$$\begin{aligned} A_{ij} &= \left[4 k_B T \left(\frac{5\eta}{3} - \zeta\right) \frac{G_{ij}}{d_i d_j}\right]^{1/2}, \\ B_{ij} &= \left[4 k_B T \left(\frac{5\eta}{3} + 8\zeta\right) \frac{G_{ij}}{d_i d_j}\right]^{1/2}. \end{aligned} \tag{2.20}$$

For a detailed derivation, readers are referred to Ref. [26].

From the SPH discretization of the Navier-Stokes equation, we obtain a version of DPD (namely SDPD), in which the equation of state and transport coefficients are explicitly specified *a priori*. Even though SDPD is a Lagrangian discretization of the Landau-Lifshitz-Navier-Stokes equation, there are strong similarities between the force equations (2.2) of classic DPD and that of SDPD. However, because of the top-down discretization approach, SDPD can only exhibit the continuum behavior

of the system. Small scale behaviors arise from some underlying physical process might not be captured by SDPD.

2.3 Correspondence to microscopic molecular-dynamics

We used Mori projection formalism in Sec. 2.2 to establish the correspondence between DPD parameters and thermodynamic response and transport coefficients of the Navier-Stokes equations. To establish the correspondence between DPD parameters and microdynamics, a more general projection method, namely Zwanzig projection, was adopted [46, 58, 64, 70, 134, 107].

Let the microdynamics be described by a Hamiltonian, and the variable of interest be denoted by A. Additionally let the initial state of the Hamiltonian system be the point z on the phase space. The evolution of the dynamics of A is $\frac{dA(t,z)}{dt} = e^{iLt} iLA(0,z)$, where $\mathcal{L}$ is the Liouville operator, and e^{iLt} is the time evolution operator.

Similar to Mori projection in Sec. 2.2, the goal of Zwanzig projection is to extract the relevant component from the system. This can be expressed mathematically with a projection operator $\mathcal{P}$ such that $\mathcal{P}\{\,\cdot\,\} \triangleq \frac{(\cdot, A)}{(A, A)} A$. Furthermore, there is a complementary operator $\mathcal{Q}$ such that $\mathcal{Q} = 1 - \mathcal{P}$.

We can re-write the time evolution of a relevant variable A using those operators:

$$\begin{aligned}\frac{dA(t,z)}{dt} &= e^{iLt}i\mathcal{L}A(0,z) \\ &= e^{iLt}i(\mathcal{Q}+\mathcal{P})\mathcal{L}A(0,z) \\ &= e^{iLt}i\mathcal{Q}\mathcal{L}A(0,z) + e^{iLt}i\mathcal{P}\mathcal{L}A(0,z) \\ &= e^{iLt}i\mathcal{P}\mathcal{L}A(0,z) + \int_0^t d\tau e^{i\mathcal{Q}\mathcal{L}(t-\tau)}i\mathcal{P}\mathcal{L}e^{i\mathcal{Q}\mathcal{L}\tau}i\mathcal{Q}\mathcal{L}A(0,z) + e^{i\mathcal{Q}\mathcal{L}t}i\mathcal{Q}\mathcal{L}A(0,z).\end{aligned} \tag{2.21}$$

In the last step, Duhamel-Dyson identity $e^{i\mathcal{L}t} = e^{i\mathcal{L}t}\mathcal{P} + \int_0^t d\tau e^{i\mathcal{L}\tau}i\mathcal{P}\mathcal{L}e^{i\mathcal{Q}\mathcal{L}(t-\tau)} + e^{i\mathcal{Q}\mathcal{L}t}\mathcal{Q}$ was substituted. The three terms in Eq. (2.21) represent the conservative, dissipative and random components, respectively. When Eq. (2.21) was used to derive equations of motion of mesoscopic particles, Lei et al. [64] found the equations of motion has the same form as DPD's.

2.4 Parameterization

To accurately simulate a fluid with a particular set of transport coefficient, we need an appropriate set of input parameters. The typical approach is to solve an inversion problem: running trial simulations and tuning input parameters until the desired transport coefficients were achieved. Although this trial and error approach is very straight-forward and accurate, it might require many trials to find the optimal set if our initial guesses were faraway. Luckily, kinetic equations can provides a guideline for tuning the DPD parameters, albeit not accurately. For the sake of brevity I will only summary the key result and reiterate the implications here. Interested readers are referred to Ref. [84].

The main result in Ref. [84] is as follow:

$$mk_BT = \frac{A_3}{A_1(2 - A_1 n\Delta t) - A_2\Delta t}, \tag{2.22}$$

where

$$A_1 = \frac{\gamma}{md}\left[w_D\right], \quad A_2 = \frac{2\gamma^2}{m^2 d}\left[w_D^2\right], \quad A_3 = \frac{\sigma^2}{d}\left[w_R^2\right]. \tag{2.23}$$

Eq. 2.22 provides a relationship between temperature k_BT and time step ΔT will the following ramifications:

- When $\Delta t = 0$, we recover the fluctuation-dissipation theorem in Eq. 2.7, which was obtained from the equilibrium solutions of Fokker-Planck equation on the stochastic differential equation interpretation of DPD formulation.
- The temperature of the system is inversely and monotonically proportional to the time step (i.e. $k_BT \propto \Delta t^{-1}$).
- If $\Delta t > \Delta t_c = (2A_1)/(nA_1^2 + A_2)$, the denominator of 2.22 will negative. This cause the system to be unstable.
- For a stable simulation, the density can not exceed a critical density

$$n_c = (2A_1 - A_2\Delta t)/(A_1^2\Delta t) \tag{2.24}$$

The fluctuation of a DPD system is determined by the compressibility of the liquid [43]. To model a liquid with a correct thermodynamics state, the repulsive parameter (i.e. a) of a DPD system must be carefully tuned to recover the compressibility of the fluid [42]. From the virial theorem, the pressure of a DPD system can be written as

$$p = \rho k_BT + \frac{2\pi}{3}\rho^2 \int_0^{r_c} r f(r) g(r) r^2 dr, \tag{2.25}$$

where $g(r)$ denotes the radial distribution function, and rho the particle number density. As always, r_c is the cutoff radius beyond which $f(r)$ vanishes. As a simplification, we can approximate the integral, and the pressure can be approximated as

$$p = \rho k_B T + \alpha a \rho^2 (\text{with } \alpha = 0.101 \pm 0.001). \tag{2.26}$$

Given the dimensionless compressiblity is computed by $\kappa_c^{-1} = (\partial p/\partial \rho)_T / k_B T$, we can readily express the repulsive force parameters a in terms κ_c and others as

$$a = k_B T(\kappa_c^{-1} - 1.0)/2\alpha\rho. \tag{2.27}$$

To match the compressibility of a realistic fluid, the dimensionless compressibility can be computed from $\kappa_c^{-1} = \frac{[L]^3}{\rho k_B T \beta_T}$, where β_T is the isothermal compressibility of the fluid, and $[L]$ is the length scale of the DPD system. Using the known compressibility of water, $\kappa_c \approx 1/16$ when a length scale of $[L] = 0.448\ nm$ is assumed. This leads to the approximation $a = 75.0 k_B T/\rho$ that are used widely for the DPD repulsive parameter. An overestimated κ_c would lower the DPD fluid's sound speed $(\partial p/\partial \rho)_T$.

The viscosity and particle self-diffusion coefficient of a DPD fluid can be expressed as a function of the the dissipative coefficient γ from Eq.2.2. As demonstrated by the Irving-Kirkwood formula [124], the disspative contribution to the stress is due to the explicit friction force between particles moving on different streamlines, which is the viscosity. On the other hand, the kinetic contribution is due to particles diffusing across streamlines, which is related to the particle self-diffusion. Assuming a uniform particle density, i.e. $g(r) = 1$, Groot and Warren derived both contributions in Ref.

[42]:

$$\eta^D = \frac{2\pi\gamma\rho^2}{15}\int_0^\infty drr^4 w_D(r), \tag{2.28}$$

$$\eta^K = \rho\left\langle \mathbf{v}^2\right\rangle \tau/3, \tag{2.29}$$

where $\frac{1}{\tau} = \frac{4\pi\gamma\rho}{3}\int_0^\infty drr^2 w_D(r)$ is the mean time between collisions. For any specific wight function $\omega_D(r)$, one can compute η^D and η^K easily. In general, the integrals can be approximated by polynomials, and the viscosity contributions can be expressed as [72]

$$\eta^D = \frac{16\pi\gamma\rho r_C^5}{5(s+1)(s+2)(s+3)(s+4)(s+5)}, \tag{2.30}$$

$$\eta^K = \frac{3k_BT(s+1)(s+2)(s+3)}{16\pi\gamma r_C^3}. \tag{2.31}$$

where s is the exponent in the force weighting function 2.3. The total viscosity is simply the sum of η^D and η^K. Therefore, the viscosity and particle self-diffusion coefficient of a DPD system can be tuned with dissipative coefficient γ.

Since $\left\langle \mathbf{v}^2\right\rangle = 3k_BT$ and $\nu^K = \rho D/2$, the diffusion coefficient can be approximated as well:

$$\begin{aligned} D &= \frac{1}{3}\int_0^\infty dt\left\langle \mathbf{v}_i(0)\cdot\mathbf{v}_i(t)\right\rangle \\ &= \tau k_BT \\ &= \frac{3k_BT(s+1)(s+2)(s+3)}{8\pi\gamma\rho r_C^3} \end{aligned} \tag{2.32}$$

Chapter Three

Acceleration with Spatial Decomposition and Graphical Processing Units

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 Figure 3.1, 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 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) [68] 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

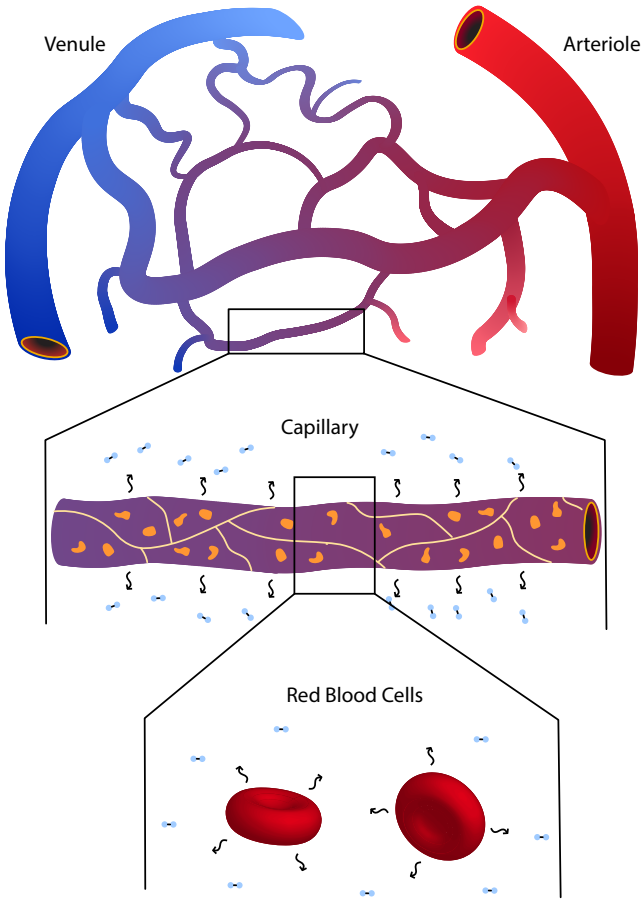


Figure 3.1: An illustration of oxygen release in the microvascular bed. RBCs transverse from oxygen-rich arterioles to oxygen-scarce venules through the connecting capillary beds. In each segment of the fenestrated capillary, oxygen and other nutrients diffuse into the surrounding tissue through the 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 chapter serves as an enabling technology to this objective.

of the Navier-Stokes equation [42]. 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 [67], platelet aggregation [100], cell sickling [66], and polymer self-assembly [68, 74, 123]. The newly developed transport DPD (tDPD) [75] 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 a RBC performs its biological function. A coarse-grained DPD membrane model for RBC was pioneered by Pivkin *et al.* [99]. 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.* [29] 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 [40, 14, 117, 80, 2, 101, 54, 103]. Pushing for the peak performance on each of the 18,688 GPUs on the Titan supercomputer, Tang *et al.* [108] 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.

For the spatial acceleration, 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 chapter is structured as follows: In Section 3.1 and 3.2, we review the classic DPD and tDPD frameworks and present a short summary of the RBC membrane model. In Section 3.4, we present model implementations and algorithmic innovations. In Section 3.5, we validate the code with verification cases. In Sections 3.6, we demonstrate the efficiency of the code by running benchmarks cases simulating pure tDPD fluids and RBC suspensions. In Section 3.7, we further demonstrate the capability of the software with a realistic microfluidic channel simulation. We summary the chapter in Section 3.8.

3.1 Formulation of Transport Dissipative Particle Dynamics

As explained in Section 2.1, 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 tracts the concentrations of all species in the volume represented by this particle. A concen-

tration gradient induces the flux of chemicals between pairs of particles, in a similar fashion to heat transfer.

As a reminder, the movements of particles can be described by the following set of stochastic ordinary differential equations (ODEs) [49]:

$$\frac{d\mathbf{r}_i}{dr} = v_i, \tag{3.1}$$

$$\frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i = \sum_{j \neq i} (\mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R), \tag{3.2}$$

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 [42]:

$$\mathbf{F}_{ij}^C = \alpha_{ij}\, \omega_C(r_{ij})\, \mathbf{e}_{ij}, \tag{3.3}$$

$$\mathbf{F}_{ij}^D = -\gamma_{ij}\, \omega_D(r_{ij})\, (\mathbf{e}_{ij} \cdot \mathbf{v}_{ij})\, \mathbf{e}_{ij}, \tag{3.4}$$

$$\mathbf{F}_{ij}^R = \sigma_{ij}\, \omega_R(r_{ij})\, \xi_{ij}\, \Delta t^{-1/2}\, \mathbf{e}_{ij}, \tag{3.5}$$

where $\mathbf{e}_{ij} = \mathbf{r}_{ij}/r_{ij}$ is the unit vector between 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 [42]. 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 [27]:

$$\sigma_{ij}^2 = 2\, k_B\, T\, \gamma_{ij}, \qquad \omega_D(r_{ij}) = \omega_R^2(r_{ij}), \tag{3.6}$$

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 [129, 61]. It can be described by the following equation:

$$\frac{dC_i}{dt} = Q_i = \sum_{j \neq i} (Q_{ij}^D + Q_{ij}^R) + Q_i^S, \tag{3.7}$$

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 denotes Fickian and random fluxes, respectively, and are given by [75]

$$Q_{ij}^D = -\kappa_{ij}\, \omega_{DC}(r_{ij})\, (C_i - C_j), \tag{3.8}$$

$$Q_{ij}^R = \epsilon_{ij}\, \omega_{RC}(r_{ij})\, \Delta t^{-1/2}\, \zeta_{ij}, \tag{3.9}$$

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 [93]

$$\epsilon_{ij}^2 = m_s^2\, \kappa_{ij}\, \rho\, (C_i + C_j), \qquad \omega_{DC}(r_{ij}) = \omega_{RC}^2(r_{ij}), \tag{3.10}$$

where ρ is the tDPD particle density.

It is easy to see the parallel between equations (3.6) and (3.10). 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 $m_s \ll m$ [75]. 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 [42]

$$D^\zeta = \frac{3k_BT}{4\pi\gamma\rho \cdot \int_0^{r_c} r^2\omega_D(r)g(r)dr}. \tag{3.11}$$

The Fickian concentration flux Q_{ij}^D depends on D^F entirely, which can be calculated by [75]

$$D^F = \frac{2\pi\kappa\rho}{3}\int_0^{r_{cc}} r^4\omega_{DC}(r)g(r)dr. \tag{3.12}$$

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 [75].

3.2 Red Blood Cell 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 Reference [29].

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

$$U_{WLC} = \frac{k_B T h_m}{4p} \frac{\frac{h}{h_m}^2 (3 - 2\frac{h}{h_m})}{1 - \frac{h}{h_m}}, \tag{3.13}$$

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

$$U_{POW} = \begin{cases} \frac{k_p}{(m-1)h^{m-1}}, & m \neq 1, \\ -k_p \log(h), & m = 1, \end{cases} \tag{3.14}$$

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

$$\mathbf{F}_{ij}^{D} = -\gamma^{T}\mathbf{v}_{ij} - \gamma^{C}(\mathbf{v}_{ij}\cdot\mathbf{e}_{ij})\mathbf{e}_{ij}, \tag{3.15}$$

$$\mathbf{F}_{ij}^{R} = \sqrt{2k_BT}\Big(\sqrt{2\gamma^{T}}(d\mathbf{W}_{ij}^{S} - tr[d\mathbf{W}_{ij}^{S}]\mathbf{I}/3) + \frac{\sqrt{3\gamma^{C}-\gamma^{T}}}{3}tr[d\mathbf{W}_{ij}]\mathbf{I}\Big), \tag{3.16}$$

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

$$V_{area} = \frac{k_g(A^t - A_0^t)^2}{2A_0^t} + \sum_j \frac{k_l(A_j - A_{0,j})^2}{2A_{0,j}}, \tag{3.17}$$

$$V_{volume} = \frac{k_v(V^t - V_0^t)^2}{2V_0^t}, \tag{3.18}$$

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 a RBC are denoted by A^t and V^t, whereas A_0^t and V_0^t represents 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

$$V_{bending} = \sum_j k_b\Big(1 - \cos(\theta_j - \theta_0)\Big), \tag{3.19}$$

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 Reference [29].

3.3 Terminology

The research project described in this chapter lies in the intersection of computational biophysics and computer science. To be more specific, high performance computing (HPC) techniques are implemented to provide a more efficient method of solving a biophysical problem. For readers who are not familiar with particle solvers or lacks basic computer science background, this section will explain some essential HPC concepts and GPU terminologies that are relevant to the chapter.

Walltime is the actual time taken from the start of a computer simulation to the end, measured in physical time. It is conceptually similar to time measured via a stopwatch. CPU time is different from walltime; CPU time measures only the time during which the CPU is actively processing instructions.

MPI is short for Message Passing Interface. It is a communication protocol for programming parallel programs. Those programs are intended to run on distributed memory supercomputers such as compute clusters. **OpenMP**, short for Open Multi-Processing, is an API that supports shared memory multiprocessing programming. An application built with OpenMP and MPI can effectively utilize a computer cluster; OpenMP is used for parallelism within a node while MPI is used for parallelism between nodes.

Radix sort is a sorting algorithm that sorts integers. It has a low compute complexity which makes it better than most comparison-based sorting algorithms.

"**Stride of an array**" refers to the number of locations in memory between successive accesses of array elements. For array [0,1,2,3,4,5,6,7,8,9], a stride access with stride size of 2 will return array [0,2,4,6,8]. A stride access with unit stride will

return the full array [0,1,2,3,4,5,6,7,8,9].

Graphics processing unit (GPU) is a specialized computer hardware that contains up to thousands of concurrent threads. Each of those threads is much less powerful than that of a CPU, but a GPU can offer greater parallelism for specific tasks such as image processing.

A **texture (mapping) unit** is a component in GPUs. It is used in conjunction with pixel and shader units to apply texture operations to pixels. Because of its role, the memory for texture units is designed for rapid reading when reads have certain access patterns. This property makes texture memory valuable for other tasks such as scientific computing. However, the texture memory is a limited resource. Smart allocation and reuse are essential for retaining its advantage.

A **stream** is a sequence of operations that execute in issue-order on the GPU. CUDA operations in different streams may run concurrently, and those operations from different streams may be interleaved. However, the computing resource (i.e. threads) on a GPU is limited. So running multiple streams may not be efficient or technically possible.

3.4 Parallel Implementation

USER-MESO

The software package developed for our purpose is based on USER-MESO. USER-MESO[122] is a GPU-accelerated extension package to LAMMPS [101] 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 $1,024$ 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 per-timestep binary particle signatures and the Tiny Encryption Algorithm;

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.

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 Figure 3.2 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. Therefore, this data storage strategy was not implemented.

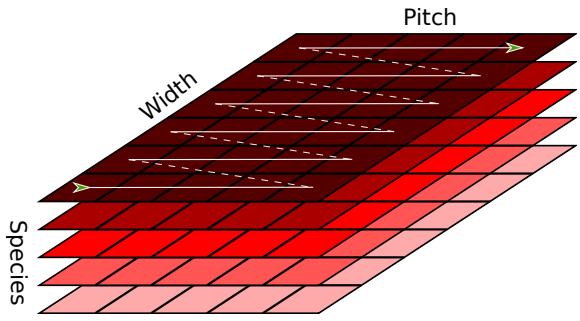


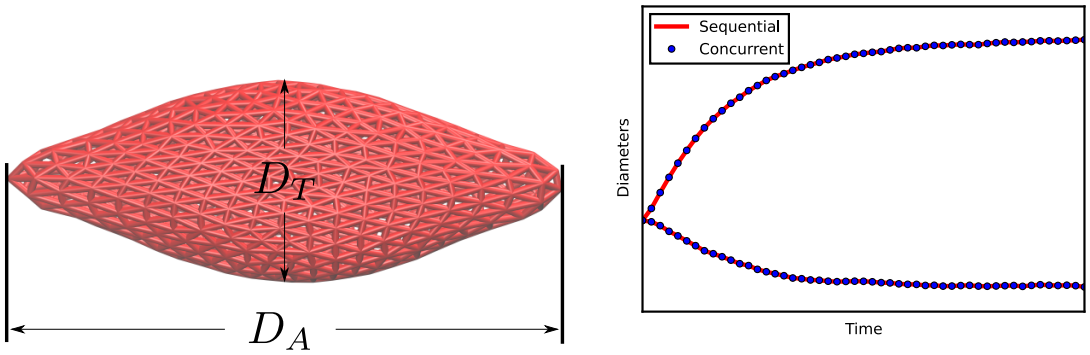
Figure 3.2: Schematic representation of 2D layered texture. Each layer holds the concentration of all particles for a particular species. The particles then wrap around to form a 2D array within each layer.

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 a RBC is split between two MPI ranks. Although the areas and volumes are accumulated in a rank-basis for the portion of a 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 Figure 3.3.



(a) (b)

Figure 3.3: A 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 matches those of the sequential version.

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 Figure 3.4. 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.

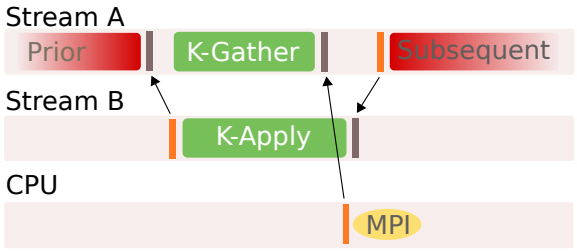


Figure 3.4: The multi-stream scheduling schematic drawing. The arrows denote dependency. *K-Apply* has to wait until the CUDA-event marking the completion of *K-Gather* is recorded. The same applies to the MPI communication.

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 Figure 3.5. 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 un-occupancy and hiding default MPI communication overhead.

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**.

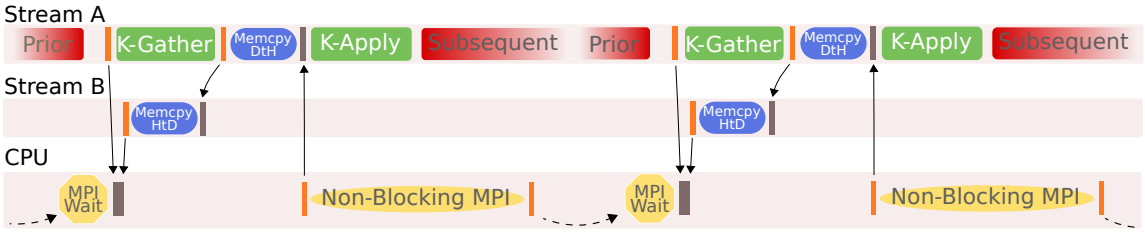


Figure 3.5: The solid arrows denote dependency within a time step, and the dashed arrows denote dependency across one time step. Async-Memcpys and Kernels are set up concurrently whenever possible to engage compute and copy engines simultaneously. Non-Blocking MPI eliminates the need for an additional device synchronization.

3.5 Code Validation

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, mimicing an infinitely long channel. The simulation parameters were chosen as $\alpha = 18.75$, $\gamma = 4.5$, $r_c = 1.58$, and $s = 0.41$. Results shown in Figure 3.6 are in good agreement with the analytical solution in [116].

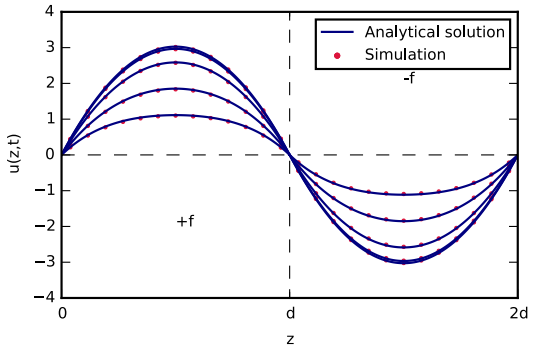


Figure 3.6: Transient velocity profile in a double Poiseuille flow. A body force f was imposed from 0 to d, and a body force $-f$ was imposed from d to $2d$. The boundary in z direction is thus zero by periodicity. Snapshots are taken at $t = 3, 6, 12, 24$ and 48, corresponding to the curves from bottom to top, to show the evolution of velocity profiles.

Transport

The diffusive property is validated by solving a one-dimensional diffusion equation in a 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 $r_{cc} = 1.58$. 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 [75]. This method also satisfies the no-slip boundary condition [66, 72].

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 [20]

$$\frac{C(t) - C_0}{C_D - C_0} = 1 - \frac{2}{R}\sum^{\infty} \frac{J_0(\lambda_n r)}{\lambda J_1(\lambda_n R)} \cdot \exp(-D\lambda^2 t), \tag{3.20}$$

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 Figure 3.7.

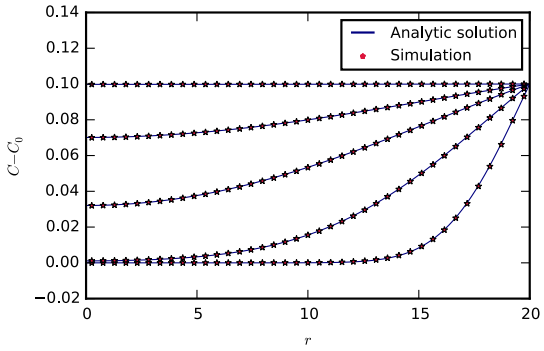


Figure 3.7: Time evolution of concentration profiles in an axial symmetric infinitely long tube. Initially the system has concentration C_0 everywhere. The boundary is kept at $C_0 + 0.1$ for $t > 0$. Snapshots are taken at $t = 1, 5, 15, 30$ and 100 to show the evolution, from the bottom curve to the top.

RBC elasticity

For validation, the RBC implementation was compared with experimental data [119] from a RBC stretching experiment via optical tweezers. The stretching was simulated by applying a constant force on few particles in opposing ends of a RBC [29]. 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 Figure 3.8. The result is identical to the one obtained using a different code by Fedosov *et al.* [29].

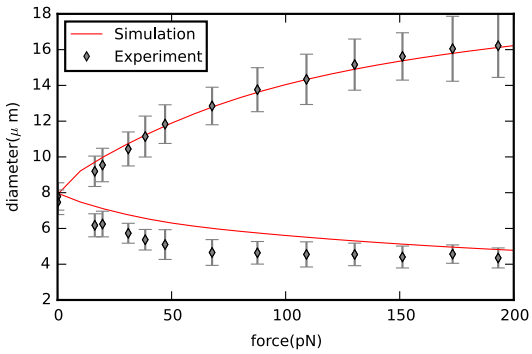


Figure 3.8: RBC elasticity validation. The top curve represents axial diameter D_A, and the bottom curve represents transverse diameter D_T. As Fedosov *et al.* [29] pointed out, the small discrepancy in transverse diameter between simulation and experiment is probably due to the optical measurement being performed from only one angle.

3.6 Benchmark

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 2.2GHz. The Kepler architecture GPU has 2,688 CUDA cores with a peak double-precision performance of 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, 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.

Single node speedup

Pure tDPD fluid The key performance metric used in this work is speedup, defined as a 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 Figure 3.9. 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.

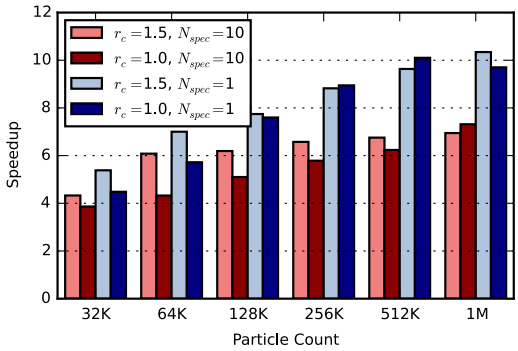


Figure 3.9: Speedup of pure tDPD fluid simulation over the CPU solver. Two different neighbor list cutoff values, $r_c = 1.0$ and 1.5, as well as two different numbers of chemical species, $N_{spec} = 1$ and 10, were considered. Neighbor lists were updated every 5 steps.

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

3.1). In this case, only one species was included. The speedup is comparable to pure tDPD fluid of similar total particle count.

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

Table 3.1: 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 8,192 translates to six RBCs. Combining with 32,768 pure fluid particles, this RBCs in fluid system has 35,768 tDPD particles total.

We also tested the performance of the code on newer GPU architectures, namely Maxwell and Pascal (Table 3.2). 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.

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

Table 3.2: 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.

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) $\cdot$ (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 $1,024$ 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 Figure 3.10. Absolute performance plateaus around 512 nodes, where the parallel overhead cancels any extra computational resources. At $1,024$ nodes, the parallel overhead and data transfer completely dominate the computation.

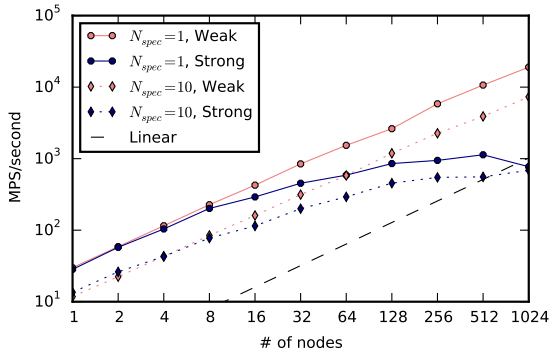


Figure 3.10: Weak scaling and strong scaling of pure tDPD fluid particles in a log-log plot.

RBC suspension Strong scaling with a system volume of $2,097,152$ is shown in Figure 3.11. 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. Switch-

ing to non-blocking reduces the execution time by approximately 25% in the case of 1,024 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.

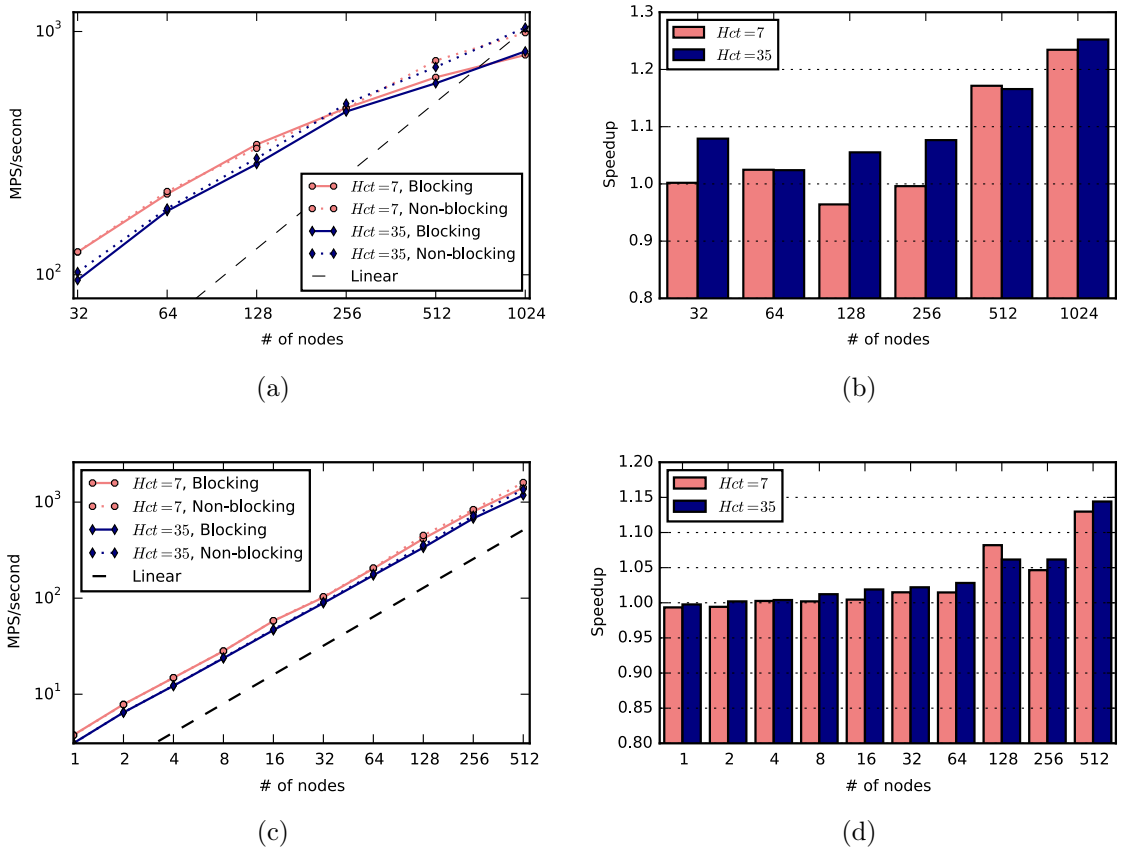


Figure 3.11: Log-log plots of (a) strong scaling and (c) weak scaling of the RBC suspension system. The speedup between non-blocking and blocking implementations for (b) strong and (d) weak scalings. Compared to the blocking versions, the non-blocking counterparts show scalability at high node counts.

3.7 Capability Demonstration

The role of RBCs in microcirculation has been investigated using microfluidic devices [48, 53, 8]. However, certain properties such as chemical concentrations are difficult

to capture experimentally with high spatial and temporal resolution.

To demonstrate its versatility, chemical diffusion from a 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 white-red intensity field shown in Figure 3.12 represents concentration gradient qualitatively. The redish tone surrounding RBCs indicates abundant presence of the chemical near the sources.

Among the $720,778$ tDPD particles in the simulation, roughly 95% represent the fluid and the solid channel wall at a density of 4 particles/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 [77]. 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.

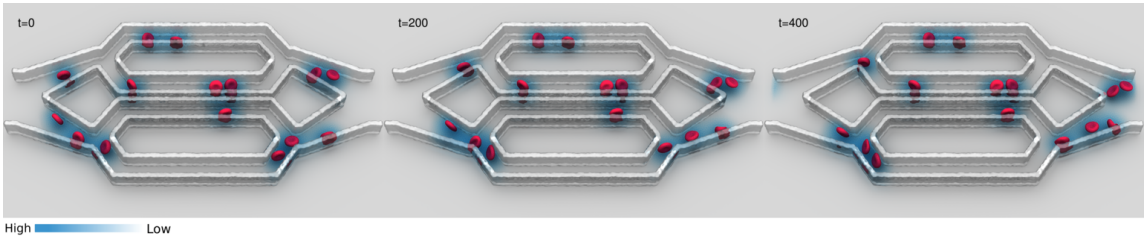


Figure 3.12: An example simulation combining RBC model and tDPD formulation is shown here. The RBC-fluid mixture flows through a microfluidic device from left to right. The white-red intensity field represents concentration gradient qualitatively. The chemical released by RBCs diffuses and dissipates in the fluid. The snapshots were taken at time 0, 200, and 400.

3.8 Summary

In this chapter, a GPU-accelerated RBC simulation package based on a tDPD adaptation of the RBC model [29] is presented. With 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 Lab. With some novel algorithms in streamlining the RBC computation, the 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 $1,024$ nodes as indicated by strong scaling benchmarks. Furthermore, we demonstrated the software's capability by simulating chemical diffusion in a RBC suspension traversing inside a microfluidic device. Incorporating the boundary resolving technique that deals with arbitrary shapes [77], 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.* [108]. We encourage researchers to adopt the user-friendly *USER*MESO in their research. The software is freely available on Github, following the link https://github.com/AnselGitAccount/USERMESO-2.0

Supplementary

For interested readers, example simulations are included in the program source code package which can be downloaded from the Github repository https://github.com/AnselGitAccount/USERMESO-2.0 . 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 section 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`.

Chapter Four

Supervised parallel-in-time algorithm for long-time Lagrangian simulations of stochastic dynamics: Application to hydrodynamics

4.1 Introduction

The lack of an efficient and stable numerical method remains a serious bottleneck for long-time simulations. Time parallel integration methods can break the bottleneck by decomposing the time domain and solving these subdomains parallel-in-time. Since the inception of time parallel integration methods over 50 years ago [92], many have improved and proposed new parallel-in-time methods. Parareal, proposed by Lions et al. [83], is a popular parallel-in-time scheme and has demonstrated its versatility in various applications. It follows a predictor-corrector paradigm in which the prediction is carried out with an inexpensive solver while the correction is performed with an expensive but parallelizable solver. The prediction is then corrected over iterations, resulting in a refined solution, which then acts as an initial condition in time for the next iteration. The traditional Parareal scheme has been applied to a wide range of problems, including fluid-structure interactions [28], reservoir simulation[39], approximation to Navier-Stokes equations [34], turbulent plasma flow [109], and even to fractional partial differential equations [128]. While most applications employ continuum solvers, a few utilize other methods such as particle solvers [6, 15, 3, 118, 37, 63]. Nevertheless, in most published works the fine and coarse propagators solve the same governing equation in an umbrella model, but with different spatial-temporal discretizations. There are a few exceptions in which the coarse propagator operates on a simplified mathematical model [9, 22, 45, 3]. However, the acceleration of Lagrangian stochastic particle solvers employing parallel-in-time strategy has yet to be developed.

We propose a **S**upervised **P**arallel-in-time **A**lgorithm for **S**tochastic **D**ynamics (`SPASD`), which aims to significantly accelerate stochastic Lagrangian solvers for long-time simulations. Based on the bottom-up coarse-graining philosophy, stochastic

particle models such as dissipative particle dynamics (DPD) converge to continuum macroscopic models in the scale limit [24, 135]. The macroscopic system can then serve as a predictor to supervise the high-dimensional stochastic Lagrangian simulation. Even though the governing equations of the macroscopic model, generally in the form of partial differential equations, are different from those of the microscopic model, the macroscopic model can capture the correct mean-field behavior of the microscopic system in the continuum limit. In particular, an inexpensive continuum solver, also known as the coarse propagator, solves the macroscopic model in serial, and an expensive but parallelizable solver, also known as the fine propagator, resolves the molecular details by performing stochastic microscopic simulations. In the examples, we will show how the coarse propagator can provide a rough prediction of the mean-field hydrodynamics that supervises the fine propagator in the time domain, as well as how the fine propagator corrects the prediction iteratively. For demonstration, we employ DPD to model the stochastic microscopic dynamics and the Navier-Stokes equations to obtain mean-field behavior in the continuum [24, 135]. It is worth noting that `SPASD` is a scalable and stable parallel-in-time algorithm applicable to many popular stochastic Lagrangian solvers, e.g., MD, DPD, smoothed DPD and Langevin dynamics. Most importantly, the strong scaling of `SPASD` does not plateau like conventional spatial decomposition methods, and `SPASD` can be implemented in conjunction with spatial decomposition methods to achieve further speedup.

Considering `SPASD` as a multiscale method, one could argue that `SPASD` resembles methods such as multigrid, heterogeneous multiscale method, and equation-free approach. However, there are clear distinctions. Categorically, `SPASD` belongs to the class of concurrent multiscale modeling, for which the macro- and micro-model are solved concurrently. Structurally, `SPASD` echoes the extended multigrid methods

(EMGM) [13], in which the effective problems are solved with different types of models corresponding to different levels of physical representation. To bridge the scales, EMGM employs techniques that are similar to `SPASD` - interpolation, relaxation and restriction, and relies on iterative refinement for convergence. However, in contrast with `SPASD` , EMGM is strictly serial in time. There have been some attempts for developing parallel-in-time multigrid methods [50], but the extension to EMGM has yet to be properly explored to the best of our knowledge.

The heterogeneous multiscale method (HMM) views the connection between the micro- and macro-model from a different perspective. In HMM, one has to first guess the form of the macro-model, which can be incomplete [125], e.g. missing the true constitutive law in complex materials. The key is to extract the required information from the micro-model to solve the incomplete macro-model. As expected, the macro-state provides the constraints for setting up the micro-state. In this regard, the difference between HMM and `SPASD` is easy to spot: the macro-model is complete and serves as a predictor in `SPASD` . In particular, guessing the wrong values of the macro-model parameters in `SPASD` is acceptable because the ramification is corrected by the algorithm, whereas guessing the wrong form of the macro-model is likely going to lead to wrong results in HMM [125].

Another well-known multiscale methodology is the equation-free approach (EFA). EFA is intended to be used when a macroscopic description is unavailable in closed form [55]. In EFA, microscopic simulations are performed only in small spatial domains over a short time period, commonly known as patches. The evolution of the macroscopic fields is then macroscopically interpolated across the patches. In contrast, microscopic simulations in `SPASD` are performed concurrently on the entire temporal domain. Furthermore, we use a macro-model to best describe the complex microscopic dynamics at a coarse level. Due to the reduced representation of the

true dynamics, the macro-model has a closed-form, and it is solvable with a fast solver.

In conclusion, treating the macro-model merely as a predictor is unique to SPASD . That is to say, the macro-model can be inaccurate and inconsistent with the microscale description. Because the parallel-in-time algorithm anticipates the deviation from the true (microscopic) dynamics, SPASD can recover the true dynamics, as demonstrated in this work.

The remainder of this chapter is organized as follows. In section 4.2 we describe SPASD and provide an analysis of its theoretical speedup. In Section 4.3 we demonstrate the implementation of SPASD for a stochastic Lagrangian simulation supervised by its mean-field approximation. Subsequently, we present quantitative results including accuracy, convergence rate and parallel efficiency for an one-dimensional time-dependent problem. Furthermore, we apply SPASD in a two-dimensional cavity flow and analyzed the results. Lastly, the chapter ends with a brief summary and discussion in Section 4.4.

4.2 Algorithm

Given that SPASD is inspired by the traditional Parareal algorithm [83], we will first briefly summarize the traditional Parareal and then introduce the proposed algorithm, SPASD. The speedup offered by SPASD is determined by the complexity of projection, mapping and filtering operations as described below. In order to retain generality, we present a theoretical speedup in which the walltimes of those operations are represented symbolically. The theoretical speedup is then compared to the

speedup of conventional spatial decomposition method in the context of distributed computing.

4.2.1 Traditional Parareal Algorithm

For the sake of clarity, let us assume that we have reduced the original problem to the following ordinary differential equation (ODE): $du/dt = f(u),\ t \in [0, T_{\text{final}}]$; see Table 4.1 for the notation. Let U_k^n be an approximation to the exact solution $u(n\Delta T)$ at time $n\Delta T$ in k^{th} iteration, where ΔT denotes the length of a time subdomain. $N = T_{\text{final}}/\Delta T$ is the number of time subdomains. Let $\mathcal{F}$ be the expensive fine propagator that accurately approximates $u(t)$, and $\mathcal{G}$ be a less accurate but inexpensive coarse propagator. Δt_f and Δt_c represent the time step of the fine and coarse propagators, respectively. We note that, although Δt_c is usually equal to the length of a time subdomain ΔT, it can be less than ΔT as demonstrated in later sections, i.e., $\Delta t_c \leq \Delta T$. Summarized in Algorithm 2, the traditional Parareal algorithm [83] places few constraints on the fine and coarse propagators. Typically, the coarse propagator employs the same numerical scheme as the fine solver, but with a coarser spatio-temporal discretization. Alternatively, they can be different numerical schemes with distinct convergence properties [22, 87]. Nevertheless, both the fine and coarse propagators operate on the same underlying governing equations in most applications.

Algorithm 2 Traditional Parareal algorithm

1: Initialization:
Compute initial iteration with Coarse propagator: $U_0^{n+1} = \mathcal{G}(U_0^n)$ for all $0 \leq n \leq N$.
2: Assume sequence $\{U_k^n\}$ is known for some $0 \leq n \leq N$ and $k \geq 0$:
Correction:
(a) Advance with Coarse propagator in serial: $\mathcal{G}(U_k^n)$ for all $0 \leq n \leq N-1$.
(b) Advance with Fine propagator in parallel: $\mathcal{F}(U_k^n)$ for all $0 \leq n \leq N-1$.
(c) Compute correction: $\delta_k^n = \mathcal{F}(U_k^n) - \mathcal{G}(U_k^n)$ for all $0 \leq n \leq N-1$.
Prediction:
Advance with Coarse propagator in serial: $\mathcal{G}(U_{k+1}^n)$ for all $0 \leq n \leq N-1$.
Refinement:
Combine the correction and prediction terms: $U_{k+1}^{n+1} = \mathcal{G}(U_{k+1}^n) + \mathcal{F}(U_k^n) - \mathcal{G}(U_k^n)$
3: Repeat *Step 2* to compute U_{k+2}^n for all $1 \leq n \leq N$ until a termination-condition is satisfied.

4.2.2 Supervised Parallel-in-time Algorithm for Stochastic Dynamics (SPASD)

In order to accelerate particle simulations, we use a mean-field approximation as the macroscopic model whose underlying governing equations are different from those of the microscopic model. Summarized in Algorithm 3 and represented graphically in Fig. 4.1, propagators in SPASD solve their respective sets of governing equations reflecting the model. As a consequence, a solution representing the macroscopic state is different from the one representing the microscopic state. A macroscopic solution from a coarse propagation can be mapped to a microscopic state via a mapping operation (denoted by the mapping operator $\mathscr{R}$). Conversely, a macroscopic solution can be obtained from a microscopic state via a projection operation (denoted by the projection operator $\mathscr{P}$). The projection and mapping processes are the core challenges

Table 4.1: Notation for selected variables.

ΔT	Length of a time subdomain
Δt_c	Time step of a coarse propagator
Δt_f	Time step of a fine propagator
K	Number of iterations
N	Number of time subdomains
$\mathcal{F}$	Fine propagator
$\mathcal{G}$	Coarse propagator
$\mathscr{F}$	Filtering operator
$\mathscr{R}$	Mapping operator
$\mathscr{P}$	Projection operator
$\mathrm{T}_{\mathrm{SD}}^{\mathrm{serial}}$	Total walltime of a serial simulation using conventional SD
$\mathrm{T}^{\mathtt{SPASD}}$	Total walltime of a simulation using `SPASD`
T^{f}	Walltime taken by fine propagations per iteration
T^{c}	Walltime taken by coarse propagations per iteration
$\mathrm{T}^{\mathscr{F}}$	Walltime taken by filtering operations per iteration
$\mathrm{T}^{\mathscr{R}}$	Walltime taken by mapping operations per iteration
$\mathrm{T}^{\mathscr{P}}$	Walltime taken by projection operations per iteration
τ^{f}	Walltime of one fine propagation
τ^{c}	Walltime of one coarse propagation
$\tau^{\mathscr{F}}$	Walltime of one filtering operation
$\tau^{\mathscr{R}}$	Walltime of one mapping operation
$\tau^{\mathscr{P}}$	Walltime of one projection operation

for virtually all types of multiscale algorithms. There is no universal method bridging the scales, and the realization of those processes should be problem-dependent. During our numerical experiments, we tested some commonly implemented methods in multiscale such as weighted kernel sampling, linearly interpolation, and nearest neighbor interpolation. For our purpose, we found that the choice of method does not impact the refined results, which is partially due to the fact that the information lost in those processes can be regained by the fine propagator.

Noise filtering, denoted by operator $\mathscr{F}$, is crucial to the success of `SPASD` because in stochastic simulations the noise would accumulate with time if left unattended. The accumulation of noise will cause the refined solution to diverge from the true solution, as demonstrated in Section 4.3.1.1.

Algorithm 3 `SPASD`

1: Initialization:
   Compute initial iteration with Coarse propagator: $U_0^{n+1} = \mathcal{G}(U_0^0)$ for all $0 \leq n \leq N$.
2: Assume sequence $\{U_k^n\}$ is known for some $0 \leq n \leq N$ and $k \geq 0$:
   Correction:
   (a) Filter solution to remove noise: $\mathscr{F}\{U_k^n\}$
   (b) Advance with Coarse propagator in serial: $\mathcal{G}(\mathscr{F}\{U_k^n\})$ for all $0 \leq n \leq N-1$.
   (c) Map the macroscopic state to microscopic space: $\mathscr{R}\{U_k^n\}$
   (d) Advance with Fine propagator in parallel: $\mathcal{F}(\mathscr{R}\{U_k^n\})$ for all $0 \leq n \leq N-1$.
   (e) Project the microscopic state to macroscopic state: $\mathscr{P}\{\mathcal{F}(\mathscr{R}\{U_k^n\})\}$
   (f) Compute the correction: $\delta_k^n \equiv \mathscr{P}\{\mathcal{F}(\mathscr{R}\{U_k^n\})\} - \mathcal{G}(\mathscr{F}\{U_k^n\})$ for all $0 \leq n \leq N-1$.
   Prediction:
   Advance with Coarse propagator in serial: $\mathcal{G}(\mathscr{F}\{U_{k+1}^n\})$ for all $0 \leq n \leq N-1$
   Refinement:
   Combine the correction and the prediction terms:
   $U_{k+1}^{n+1} = \mathcal{G}(\mathscr{F}\{U_{k+1}^n\}) + \mathscr{P}\{\mathcal{F}(\mathscr{R}\{U_k^n\})\} - \mathcal{G}(\mathscr{F}\{U_k^n\})$ for all $0 \leq n \leq N-1$.
3: Repeat *Step 2* to compute U_{k+2}^n for all $1 \leq n \leq N$ until a termination-condition is satisfied.

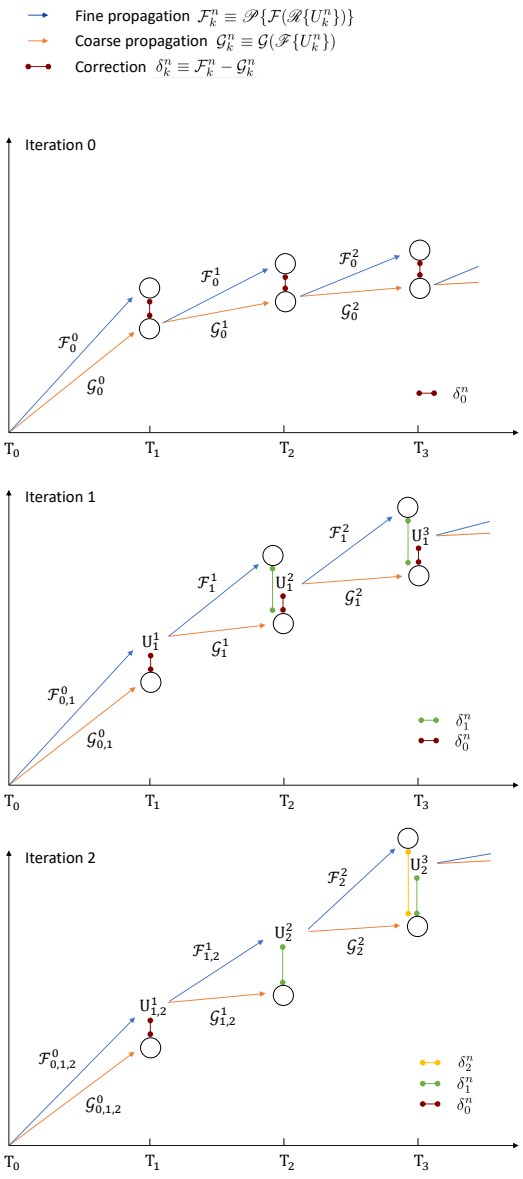


Figure 4.1: A graphical representation of our proposed algorithm `SPASD`. Unlike fine $\mathcal{F}$ and coarse $\mathcal{G}$ propagations in the traditional Parareal algorithm, $\mathcal{F}$ and $\mathcal{G}$ in `SPASD` encompass other operations, i.e., mapping, projection, filtering. The mapping and projection operators in $\mathcal{F} \equiv \mathscr{P}\{\mathcal{F}(\mathscr{R}\{U_k^n\})\}$ serve to link the two scales in the multiscale context, while the filtering operator in $\mathcal{G} \equiv \mathcal{G}(\mathscr{F}\{U_k^n\})$ is designed to suppress the stochastic noise in the model. In Iteration 1, U_1^1, U_1^2 and U_1^3 are computed from their respective $\mathcal{F}_k^n$ and $\mathcal{G}_k^n$. Because the initial state is fixed, $\mathcal{F}_0^0$ and $\mathcal{F}_1^0$ are identical. Consequentially, U_1^1 is equivalent to U_2^1 in Iteration 2.

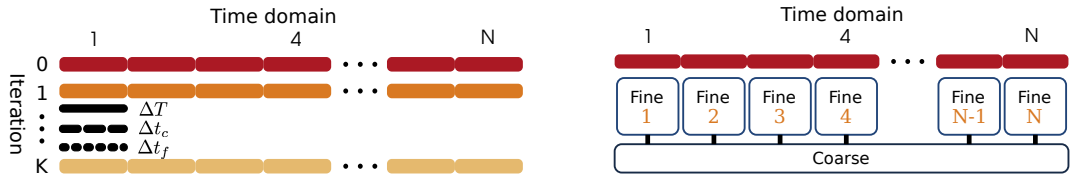


Figure 4.2: (a) A schematic representation of the temporal decomposition in SPASD. The entire time domain is divided into N subdomains, each of which is represented by a segment. As described in Algorithm 3, the first iteration is the initialization stage, which involves only the coarse propagation. (b) Each fine propagator performs computation on each subdomain, and only one coarse propagator is used for the entire time domain. Let τ^c and τ^f denote the walltime taken by each coarse and fine propagation, respectively. For the sequential coarse propagation, the total walltime spent on one iteration is $N \cdot \tau^c$. Since the fine propagators can be executed concurrently, the walltime taken by fine propagation is simply τ^f for one iteration. In summary, the total walltime spent on each iteration is thus $N \cdot \tau^c + \tau^f$ and $K \cdot (N \cdot \tau^c + \tau^f)$ for K iterations.

As with all iterative schemes, the refinement process stops when a termination-condition is satisfied. Because the magnitude of the stochastic noise varies from application to application, the idea of a universal condition is imprudent. We thus propose a practical condition that can be tuned based on the application:

$$\frac{\left\| \mathcal{G}(\mathscr{F}\{U^n_{k+1}\}) - \mathcal{G}(\mathscr{F}\{U^n_k\}) \right\|_{l_1}}{\left\| \mathcal{G}(\mathscr{F}\{U^n_{k+1}\}) \right\|_{l_1}} \equiv C_{\mathrm{TC}} < C_{\mathrm{Tolerance}}, \tag{4.1}$$

where $C_{\mathrm{Tolerance}}$ is a user-defined tolerance, which should be determined by considering the magnitude of the intrinsic stochastic noise in the system.

4.2.3 Theoretical Speedup

The total walltime can be expressed as $\mathrm{T}^{\texttt{SPASD}} = K \cdot (\mathrm{T}^{\mathrm{f}} + \mathrm{T}^{\mathrm{c}} + \mathrm{T}^{\mathscr{F}} + \mathrm{T}^{\mathscr{R}} + \mathrm{T}^{\mathscr{P}})$, where K is the number of iterations to convergence. The walltimes taken by the coarse and fine propagators per iteration are denoted by T^c and T^f, respectively. The rest of

the terms $\mathrm{T}^{\mathscr{F}}$, $\mathrm{T}^{\mathscr{R}}$ and $\mathrm{T}^{\mathscr{P}}$ denote walltime spent on noise-filtering, up-scaling and down-scaling, respectively. In accordance with the notation in Fig. 4.2, let N be the number of concurrent fine propagations. T^{c} and T^{f} can therefore be expressed as

$$\mathrm{T}^{\mathrm{c}} = N \cdot \tau^{\mathrm{c}}; \tag{4.2}$$

$$\mathrm{T}^{\mathrm{f}} = \tau^{\mathrm{f}}, \tag{4.3}$$

where τ^{f} is the walltime taken by the fine solver for one time-subdomain, and τ^{c} is the walltime taken by the coarse solver for the same time-subdomain. $\mathrm{T}^{\mathscr{F}}$, $\mathrm{T}^{\mathscr{R}}$ and $\mathrm{T}^{\mathscr{P}}$ can also be expressed similarly as $\mathrm{T}^{\mathscr{F}} = N \cdot \tau^{\mathscr{F}}$, $\mathrm{T}^{\mathscr{R}} = N \cdot \tau^{\mathscr{R}}$ and $\mathrm{T}^{\mathscr{P}} = N \cdot \tau^{\mathscr{P}}$ respectively, where τ is the walltime taken by each corresponding operation. The total walltime can thus be written as

$$\mathrm{T}^{\texttt{SPASD}} = K \cdot \left[\tau^{\mathrm{f}} + N \cdot (\tau^{\mathrm{c}} + \tau^{\mathscr{F}} + \tau^{\mathscr{R}} + \tau^{\mathscr{P}})\right]. \tag{4.4}$$

In contrast, the same simulation with only the fine solver running in serial would take $\mathrm{T}^{\mathrm{serial}} = N \cdot \tau^{\mathrm{f}}$. When conventional domain decomposition is applied using the same number of cores, the walltime is reduced to

$$\mathrm{T}^{\mathrm{serial}}_{\mathrm{SD}} = \beta \cdot N \cdot \tau^{f}, \tag{4.5}$$

where $N^{-1} \leq \beta \leq 1$. For methods that scale perfectly linearly, β is equal to N^{-1}. In the case that additional resources do not result in speedup, β is equal to 1. Comparing the walltime of `SPASD` in Eq. (4.4) with conventional domain decomposition in Eq. (4.5), `SPASD` offers better speedup when

$$\beta > \frac{K \cdot \left[\tau^{\mathrm{f}} + N \cdot (\tau^{\mathrm{c}} + \tau^{\mathscr{F}} + \tau^{\mathscr{R}} + \tau^{\mathscr{P}})\right]}{N \cdot \tau^{\mathrm{f}}}. \tag{4.6}$$

Typically, $\tau^{\mathrm{f}} >> \tau^{\mathrm{c}}$, and both τ^{f} and τ^{c} are much larger than $\tau^{\mathscr{F}}$, $\tau^{\mathscr{R}}$ and $\tau^{\mathscr{P}}$. Therefore, Eq. (4.6) can be reduced to

$$\beta \gtrapprox K\Big[\frac{1}{N} + \frac{\tau^{\mathrm{c}}}{\tau^{\mathrm{f}}}\Big]. \tag{4.7}$$

We note that in order to select the number of time subdomains, we must carefully consider the physics of the underlying stochastic dynamics. In particular, each particle model is associated with a set of dynamical properties, which describe the way collective behavior leads to macroscopic observables. For hydrodynamics, this is given by the characteristic time for momentum decorrelation, which can be quantified by the temporal correlation function of momentum, defined as $C(t) = \langle \mathbf{p}(t+\tau)\mathbf{p}(\tau)\rangle$, where $\mathbf{p}(t) = mass \cdot \mathbf{v}(t)$ represents the instantaneous momentum. The computed $C(t)$ of simple isothermal DPD fluid is shown in Fig. 4.3, normalized by the value of $C(0)$. The decay of $C(t)$ signifies momentum decorrelation over time, and correspondingly we should use a suitable ΔT in our Parareal algorithm so that momentum is sufficiently decorrelated. From our benchmark tests, $C(t) = 10^{-4}$ provides a sufficient decorrelation of momentum and leads to accurate results. Therefore, in the present work, we use $\Delta T = 10$, where the value of $C(t)$ is approximately 10^{-4} according to Fig. 4.3.

4.2.4 Methods

`SPASD` offers great versatility and robustness in terms of propagator choices. As a framework targeting particle methods, `SPASD` does not limit the type of particle solvers; that is to say, any particle solver can be accelerated with `SPASD`. For demonstration, we model mesoscale hydrodynamics with DPD, a stochastic Lagrangian

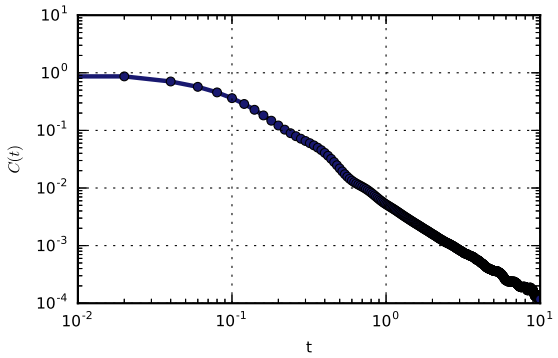


Figure 4.3: Normalized temporal correlation function of momentum, $C(t) = \langle \mathbf{p}(t)\mathbf{p}(0)\rangle$, computed from a simple DPD fluid is plotted against time. The decay of $C(t)$ signifies momentum decorrelation over time for simple isothermal DPD fluid. From our benchmark tests, $C(t) = 10^{-4}$ provides a sufficient decorrelation of momentum and leads to accurate results. Therefore, in the present work, we use $\Delta T = 10$, where the value of $C(t)$ is approximately 10^{-4}.

method. DPD can be derived rigorously from coarse-graining of atomistic dynamics [69, 73] and can recover macroscopic hydrodynamics in the continuum limit [76]. For a simple DPD fluid, a suitable low-dimensional model would be the continuum representation of fluid, namely the Navier-Stokes equations. Since there is no constraint on the discretization of the continuum equations, for simplicity, we chose the finite difference (FD) scheme as the coarse propagator.

The concurrent coupling of two independent solvers requires robust on-the-fly data transfer. For the examples shown here, we used a light weight library called Multiscale Universal Interface [121], or MUI for short, to bridge the solvers.

4.2.4.1 Dissipative Particle Dynamics

The DPD framework is explained in detail in Chapter 2.1. A common choice for the parameter s in the weighting functions (2.3) is $s = 2$, which is used for the simulations shown in this chapter. For simplicity, k_BT and the mass of a particle

are taken as the energy unit and mass unit, and their values are set to one.

4.2.4.2 Model Coupling

By applying a Mori-Zwanzig projection operator technique to a DPD system, Español [24] and Marsh et al. [85] formally derived the hydrodynamic equations of a simple DPD fluid using the Green-Kubo formulas, recovering the continuity and Navier-Stokes equations. For more details on the mathematical derivations leading to macroscopic hydrodynamic equation from DPD dynamics, we refer the interested readers to a paper by Español [24] and a book by Zwanzig [135]. Therefore, the continuity and Navier-Stokes equations can be considered as a mean-field representation of the DPD system, and consequently provide macroscopic solution as a rough prediction of the hydrodynamics that can be used to supervise the DPD simulations in the time domain.

For coupling the grid-based Navier-Stokes solver with the Lagrangian DPD solver, we found MUI easy to implement in the sense that codes do not need to be refactored before application. Adding only a few lines of code to existing solvers is sufficient to adhere to MUI's push-fetch workflow - data pushed by one solver is then fetched by the other. In the context of particle solvers, the relevant data are particle positions and the associated quantities. For continuum solvers, the quantities of interest are associated with nodes of the discretization grid. In a data-transfer, the sender pushes the data into a MUI Interface Layer (**IF**), which then assists the receiver to fetch and decode these data based on the topological configuration. Because topological configuration in the receiver usually do not exactly overlap with these of the particle solver, a sampler such as Gaussian kernel or nearest neighbor might be used to calculate an appropriate nodal value in the receiver. As for the linking configuration,

each of the fine propagators is attached to one **IF**, and all of those **IF**s link to the coarse propagator.

4.3 Numerical Results

In our proposed framework, the macroscopic model is a mean-field approximation to our microscopic model. As a consequence, the physical parameters in the macroscopic model cannot be calculated or deduced precisely from the microscopic model. However, the macroscopic model in `SPASD` only requires rough estimations of these physical parameters, effects of which can be iteratively "corrected" by the correction term in Algorithm 3. In the example problems below, we also demonstrate empirically that a more accurate parameter estimation in the macroscopic model leads to faster convergence. In addition, we will analyze the accuracy and convergence of `SPASD` on the example problems.

4.3.1 One-dimensional time-dependent flow

As a demonstration, we applied `SPASD` to the one-dimensional time-dependent plane Poiseuille flow with DPD and FD as fine and coarse propagators, respectively. Simulating Poiseuille flow is a well-accepted benchmark test for many Lagrangian methods including DPD because the macroscopic parameter viscosity can be computed empirically from simulations [4]. We refer to the computed viscosity as the true viscosity ν_{true}, which is incorporated in the true solution that is used in error analysis.

Macroscopically, the pressure-driven Poiseuille flow can be described by the in-

compressible Navier-Stokes equations:

$$\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = \nu\nabla^2\mathbf{u} + F, \tag{4.8}$$

where ν is the viscosity, F denotes a body-force driven by a constant pressure gradient, and $\mathbf{u}$ denotes the velocity field. The exact time-dependent solution to the Navier-Stokes equation is known analytically [116]:

$$u(x,t) = \frac{Fd^2}{8\nu}\left(1-\left(\frac{2x}{d}^2\right)\right) - \sum_{n=0}^{\infty} \frac{4(-1)^n F d^2}{\nu\pi^3(2n+1)^3} \cdot \cos\left(\frac{(2n+1)\pi x}{d}\right) \cdot \exp\left(-\frac{(2n+1)^2\pi^2\nu t}{d^2}\right), \tag{4.9}$$

The true solution $u(\nu_{\text{true}})$ can thus be readily calculated according to Eq. (4.9). For this problem, we use a periodic box with the size of $30 \times 20 \times 10$ in DPD units containing $24,000$ particles. The DPD parameters α and γ are set to 18.75 and 4.5, respectively. All particles are subject to a body-force $F = 0.1$ and a radial cutoff $r_c = 1.58$. Given these parameters, the true viscosity for this system is approximately 0.841 computed via DPD simulations.

As explained in Section 4.2.3, the momentum is sufficiently decorrelated beyond 10 time units according to the temporal correlation function $C(t)$. The communication interval between the propagators ΔT, also known as the time step of the coarse propagator, is thus set to be 10. With the time step $\Delta t_f = 0.01$, the fine propagators march forward for 1000 time steps during the same period. Because the Poiseuille flow is one-dimensional, we use a simple linear interpolation as the projection operation. To find the velocity profile, the simulation domain is divided into slabs of uniform width that are also the size of spatial discretization in the coarse propagator. We then average particle velocities in each slab. The result is a one-dimensional velocity profile that describes the macroscopic state. On the other hand, the macroscopic state is mapped to the particle system with a nearest-neighbor sampler - a

particle takes on the velocity of the nearest grid node. The coarse propagation and the filtering of stochastic noise generated by the fine propagator are carried out simultaneously in this example. The viscosity of fluid can naturally create a low-pass filtering effect [44] that damps high-frequency fluctuations as the coarse solver advances. To take advantage of the filtering effect, one time subdomain $\Delta T = 10$ is covered by 100 sequential coarse steps with an effective time step $\Delta t_c = 0.1$.

4.3.1.1 Impact of Filtering

As an iterative scheme, the convergence of `SPASD` greatly relies on the notion that, for a propagation, the stochastic noise in the model should not cause the solution to deviate far from the ensemble solution. To ensure convergence, a filtering operation must be incorporated in the algorithm. If the filtering operation is not enforced, the noise can grow in the temporal space within an iteration and will propagate across iterations. Unchecked noise growth can eventually drive the propagated solution away from the true solution. To demonstrate the impact of filtering, we compared results of the Poiseuille flow simulations without and with partial filtering. For fluids, the macroscopic model has a filtering mechanism build-in: the viscosity of fluid is a natural low-pass filter that damps high-frequency fluctuations. To take advantage of the filtering effect, a time subdomain ΔT is divided into smaller (effective) time steps Δt_c which the coarse solver uses.

To demonstrate the impact of filtering, we run a simple example without and with partial-filtering. The same effective time step, subsequently the same Courant-Friedrichs-Lewy condition, are used in both simulations to isolate the effect of filtering. For a time subdomains of 10 time units, the coarse propagator with an effective time step of 1 unit filters the solution 10 consecutive times. This approach offers

better noise reduction than the case with a subdomain size of 1 unit and an effective time step of 1 unit, which offers no filtering. The impact of filtering is apparent in this comparison: the solution in the no-filtering case in Fig. 4.4(a) is substantially nosier than the one in the partial-filtering case in Fig. 4.4(b). To quantify the deviation from the reference solution, we compute the normalized l_2 error, expressed as

$$\epsilon_{l_2} \equiv \frac{\left\| u^{\text{ref}} - u^{\text{sim}} \right\|_{l_2}}{\left\| u^{\text{ref}} \right\|_{l_2}}, \tag{4.10}$$

$$\left\| u^{\text{ref}} - u^{\text{sim}} \right\|_{l_2} = \left(\sum_i \left| u_i^{\text{ref}} - u_i^{\text{sim}} \right|^2 \cdot \Delta x \right)^{1/2}, \tag{4.11}$$

where Δx denotes the size of spatial discretization, and i is the nodal index. As shown in Fig. 4.4(c), the normalized l_2 error ϵ_{l_2} for the no-filtering case is monotonically increasing with time, which signifies the growth of noise in time within an iteration. The noise also propagates across iterations shown in Fig. 4.4(d). For the partial-filtering case, the noise accumulates at a much slower rate than the no-filtering case.

4.3.1.2 Accuracy

To demonstrate the flexibility and robustness of `SPASD`, we purposely estimated the viscosity in the macroscopic model to be 10× the true viscosity. In other words, the estimated mean-field solution $u(\nu_{\text{est}}) = u(\nu_{\text{true}})/10$. To verify the accuracy of `SPASD`, we compare the true velocity profiles with the ensemble average obtained with multiple `SPASD` runs. Because the microscopic model is stochastic in nature, it is desirable to obtain an ensemble averaged solution that is less distorted by stochastic noise. In our experiment, the simulated velocity profile is ensemble-averaged from fifty independent simulations. The resulting transient velocity profiles, as well as the

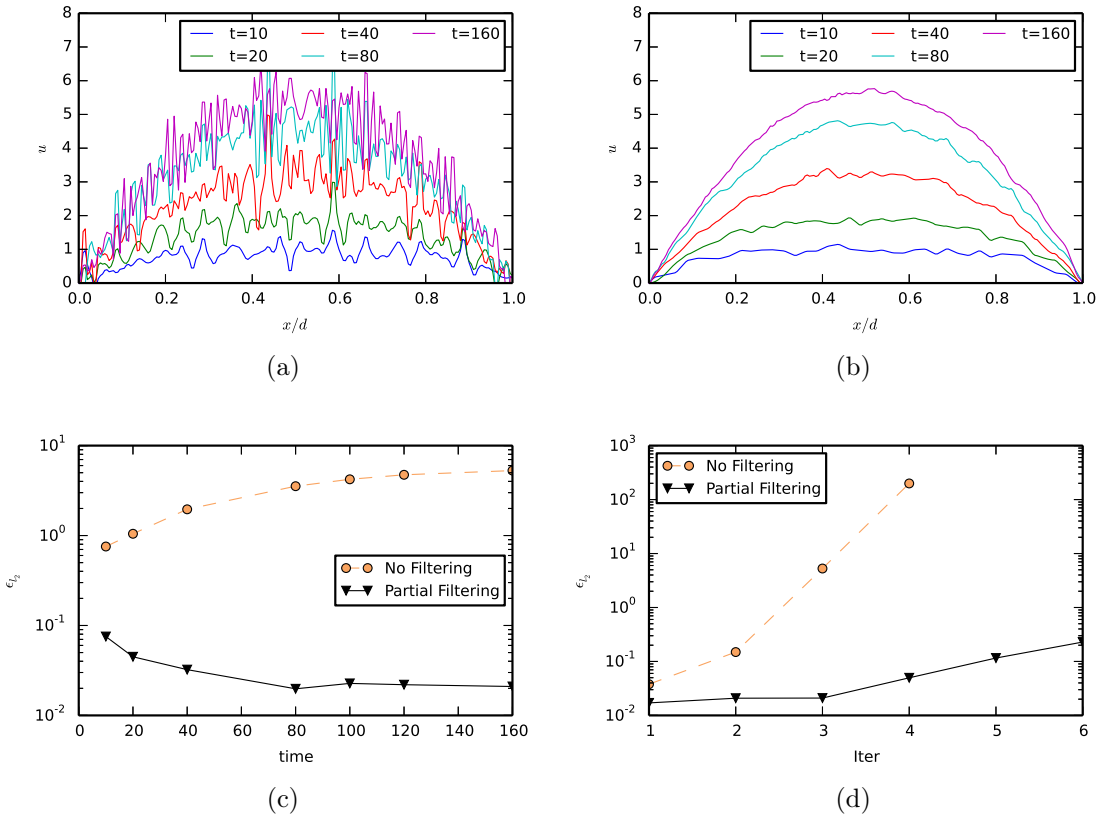
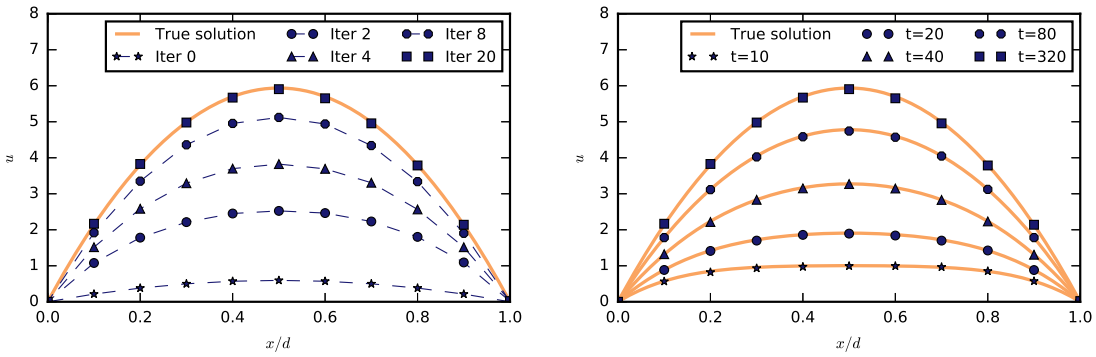


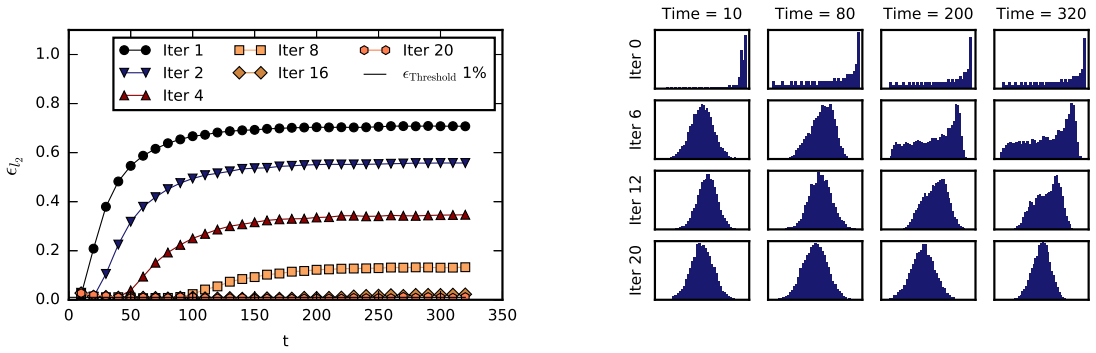
Figure 4.4: The impact of filtering is apparent when the simulations (a) without and (b) with partial filtering are compared. In the partial-filtering case where the length of a time subdomain is 10 units, the coarse propagator with an effective time step of 1 unit filters the solution 10 consecutive times. The noise is much more prominent when the length of a time subdomain is reduced to 1 unit while the effective time step is unchanged. In that case, filtering is not performed. (c) Moreover, quantified as normalized l_2 error, the noise increases monotonically with time within an iteration. (d) Across iterations, it also builds up and eventually overtakes the solution.



(a) (b)

Figure 4.5: Results of plane Poiseuille flow simulation with SPASD. (a) Evolution of steady-state velocity profile over iterations. The parameter viscosity in the low-dimensional model is 10× the true value. In that case, the velocity profile obtained with SPASD is refined and converges to the true solution iteratively. (b) Transient velocity profiles obtained with SPASD. On Iteration 20, the profiles obtained with SPASD are compared with the true velocity profiles. This demonstrates that SPASD is able to obtain correct transient solutions as well as the steady state solution.

evolution of velocity profile at steady state, match those of the true velocity profiles visually as shown in Fig. 4.5. To quantify the accuracy of SPASD , we computed the normalized l_2 error ϵ_{l_2} between the reference and simulated solutions. For the one-dimensional Poiseuille plane-flow, the reference solution is the true solution $u(\nu_{\text{true}})$ calculated according to Eq. (4.9) because the analytical form is known. We found that ϵ_{l_2} decreases over iterations as shown in Fig. 4.6(a). Given that the noise cannot be eliminated by averaging fifty solutions, we accept solutions within 1% error as a threshold denoted as $\epsilon_{\text{Threshold}}$. In this particular example, ϵ_{l_2} falls to $\epsilon_{\text{Threshold}}$ in 20 iterations. At the threshold, the error has a Gaussian distribution as shown in Fig. 4.6(b), which matches the noise distribution from the microscopic model [42]. The recovery of the distribution signifies that the SPASD preserved stochastic fluctuations of the microscopic model.



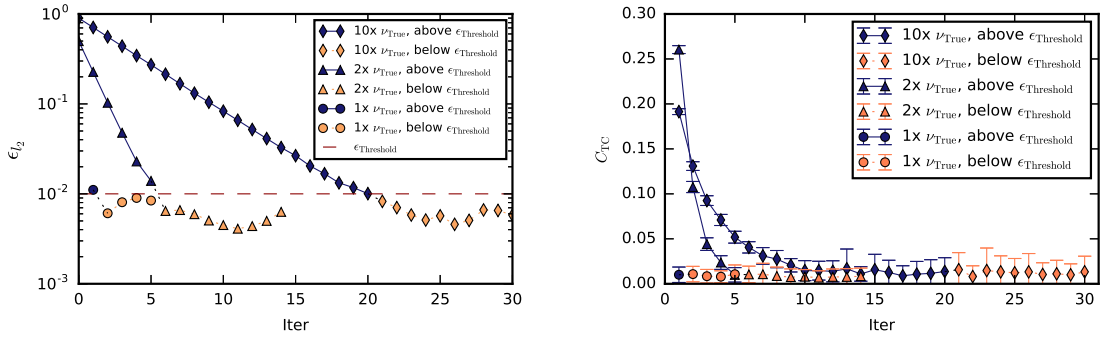
(a) (b)

Figure 4.6: Analyzing the results of plane Poiseuille flow simulation with SPASD. (a) Normalized l_2 error between simulation and true solution, denoted as ϵ_{l_2}. Because of intrinsic noise from the microscopic model, we accept solutions within 1% difference. (b) Normalized l_2 error histogram. The error histogram eventually recovers Gaussian distribution for all times, indicating convergence.

4.3.2 Convergence

Inspecting ϵ_{l_2} at $T_{\text{final}} = 320$ for all iterations in Fig. 4.7(a), SPASD achieves exponential convergence. We repeat the experiment with a more accurate macroscopic viscosity at 2× true viscosity and obtain exponential convergence as well, but only 5 iterations are needed to reach the same threshold. More importantly, ϵ_{l_2} reaches a minimum level regardless of the estimated viscosity. The minimum error is determined by many factors; in addition to the error intrinsic to Parareal, operations in SPASD such as filtering and projection introduce systematic errors that are difficult to separately quantify.

Lastly, we calculate C_{TC} from Eq. (4.1) and plot it in Fig. 4.7(b). Because of the stochastic noise, C_{TC} varies between runs. We plot the mean and 95% confidence interval for C_{TC} generated from fifty independent runs. When the parameter estimation is more accurate as in the case of 2x true viscosity, C_{TC} fluctuates mildly as indicated by the range of the confidence interval and drops sharply to a level



(a) (b)

Figure 4.7: (a) l2 error. ϵ_{l_2} at T_{final} is plotted against iterations. In addition to 10×, two experiments were performed with estimated viscosities that are 2× and 1× true viscosity. The nearly linear plot on a semi-log scale indicates exponential convergence. Moreover, when true viscosity is used, it takes one iteration to reach the error threshold. (b) Termination condition. The termination criterion C_{TC} from Eq. (4.1) is plotted. Solid navy lines correspond to iterations before the error threshold, and dotted yellow lines after the error threshold.

that satisfies the termination condition. However, when estimation is rough, the confidence interval is relatively wide without a clear boundary separating iterations above and below $\epsilon_{\text{Threshold}}$. The resulting consequence is discussed in Section 4.4.

4.3.3 Efficiency

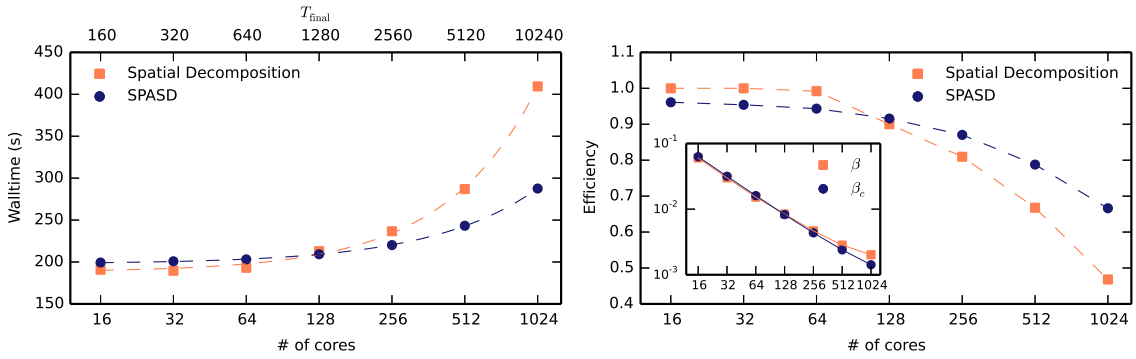
For particle simulations, spatial domain decomposition is the most utilized acceleration method because the subdomains can be easily assigned to different processors. To compare the performance of `SPASD` with conventional spatial decomposition (CSD), we apply them separately to the planer Poiseuille flow problem and then calculate the parallel efficiency.

The same simulation parameters described in Section 4.3.1 are used in this benchmark test. To avoid implementing no-slip boundaries, we simulated a reverse

Poiseuille flow [5, 71], in which two planar flows are stacked to create periodic boundaries in all directions. At the same time, the z-direction is shrunk by one half to preserve the total number of particles. For a general comparison, we simulate a single iteration, and each core was responsible for 10 time units. When the number of available cores is doubled, the final time T_{final} is also doubled.

This particular benchmark was run on nodes with AMD Opteron 6274 CPUs. For this example, the performances of CSD and `SPASD` are similar when fewer cores are available as shown in Fig. 4.8(a). As the core-count and T_{final} increase, their performances start to diverge at approximately 128 cores. From there, the walltime of CSD grows at a much faster exponential rate than that of `SPASD`. Two factors contribute to this: First, the size of a spatial subdomain in CSD is limited by the average particle distance. Second, the time spent on communication between nodes grows at a rate that nullifies the benefits of additional cores. `SPASD`, however, does not suffer from the same issues. Communication between nodes is only invoked at the end of temporal subdomain, and the duration of temporal subdomain can be adjusted at will. This makes `SPASD` a more scalable algorithm suitable for long-time simulations. For comparison, we used the speed of a single-core simulation, roughly 39 seconds per time unit, as a basis and computed the speedup of each method. The efficiencies were then calculated as the speedups per core. As expected, `SPASD` offers better parallel efficiency in the cases of large core counts, as shown in Fig. 4.8(b).

The walltime taken by the coarse propagator and associated operations $\tau^{\text{c}} + \tau^{\mathscr{F}} + \tau^{\mathscr{R}} + \tau^{\mathscr{P}}$, as well as the walltime taken by the fine propagator τ^{f}, can be readily calculated with a simple linear fit to Eq. (4.4). For this benchmark case, the walltime of the simulation using `SPASD` is described by $\text{T}^{\texttt{SPASD}} = 197.8 + 0.08783N$ where $N = T_{\text{final}}/10$. Plugging in the numerical value for τ^{f}, the walltime of simulation using CSD is reduced to $\text{T}_{\text{SD}}^{\text{serial}} = 197.8\beta N$. The scaling factor β depends on many



(a) (b)

Figure 4.8: Comparison between SPASD and conventional spatial decomposition (CSD). (a) Walltimes of simulation with CSD and simulation with SPASD. The performances of CSD and SPASD are similar when fewer cores are available. As the core-count and T_{final} simultaneously increase, the performance of SPASD starts to dominate. The walltime of CSD grows exponentially at a much faster rate than SPASD. (b) Using the walltime of a single-core simulation at different T_{final} as a basis, the speedup of CSD and SPASD can be computed. The efficiencies were then calculated as the speedup per core. The higher parallel efficiency at high core-counts indicates that SPASD is a more scalable algorithm for long-time simulations. In reference to Eq. (4.6), we also compared $\beta = \mathrm{T}_{\mathrm{SD}}^{\mathrm{serial}}/(\tau^f N)$ and $\beta_c = [\tau^{\mathrm{f}} + N \cdot (\tau^{\mathrm{c}} + \tau^{\mathscr{F}} + \tau^{\mathscr{R}} + \tau^{\mathscr{P}})]/(N \cdot \tau^f)$. Starting from 128 cores and up, β begins to flatten out while β_c continues to drop. The trend echoes with the efficiency plot and reaffirms that SPASD starts to dominate as few as 128 cores for this example.

factors including the number of parallel cores, distribution topology, and parallel-method efficiency. When few cores are available, β is approximately equal to N^{-1} as shown in Fig. 4.8(b), and CSD provides better efficiency and performance. As the number of cores increases, β decreases proportionally until around 128 cores at which `SPASD` and CSD have similar parallel efficiencies. At this core-count, β was calculated to be 0.0083, which agrees with our prediction $\beta_c \equiv K(N^{-1} + \tau_c/\tau_f) = 0.0084$ given by Eq. (4.7). From this core-count forward, the flattening of the β curve signifies the drop of CSD efficiency.

4.3.4 Two-dimensional time-dependent flow

To further demonstrate the applicability of `SPASD`, we simulated a two-dimensional cavity flow. Simulating transient cavity flow is much more complicated than the Poiseuille flow because the convective acceleration and the corner singularity induced by the cavity make it numerically challenging. However, the singularities, including corner singularity [91], crack propagation [12], and moving contact line singularity [115], in partial differential equation (PDE) models (in the limit of $\Delta x \to 0$) represent rich underlying multiscale physics, and can be captured by particle solvers, e.g., DPD or MD.

In this case, we again use DPD to simulate the lid-driven cavity flow, which is supervised in the temporal domain by the incompressible Navier-Stokes equations. These equations are solved with finite difference on a staggered grid: the pressure is stored at the center of the grid cells, and the velocities are stored at the faces of the grid cells. The second order spatial derivative is approximated with a centered difference scheme. This setup allows pressure and velocity to be solved concurrently in a two-step process. The first step is to compute an intermediate velocity by solving

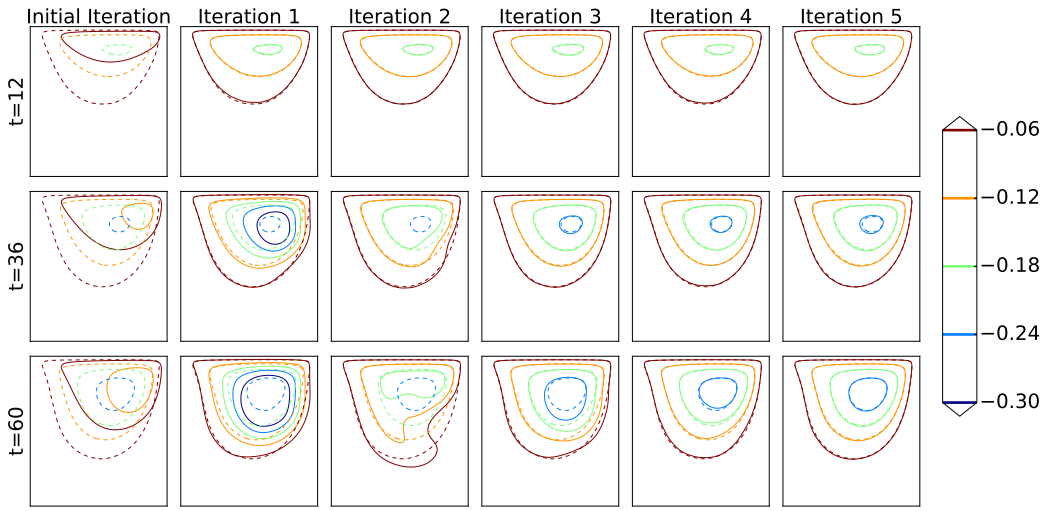
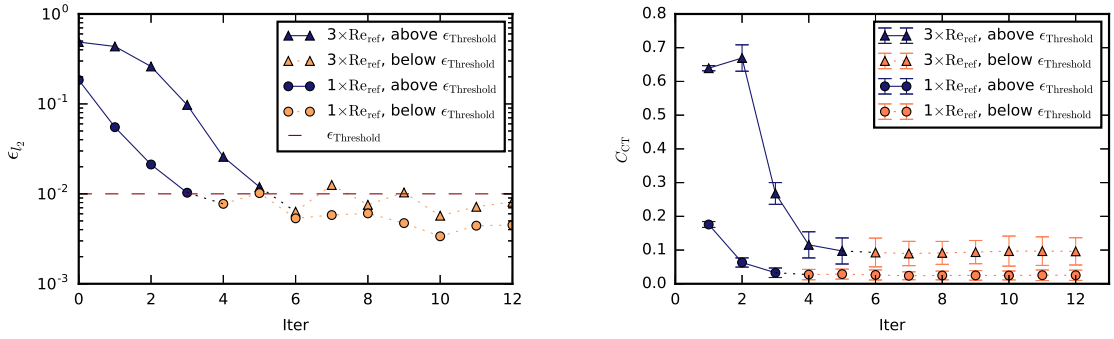


Figure 4.9: Stream function of the two-dimensional cavity flow with `SPASD`. The dashed and solid contour lines represent the reference and simulation solutions, respectively. In the initial iteration, the center of the vortex matches poorly with the reference solution. This is expected because the low-dimensional model is solved with an estimated Reynolds number $\mathrm{Re}_{\mathrm{est}}$ that is $3\times$ the reference Reynolds number $\mathrm{Re}_{\mathrm{ref}}$, and only the predictor is involved computing the flow in the initial iteration. On the third iteration, the center of the vortex converges to the reference vortex.

the momentum equation, omitting the effect of pressure. The intermediate velocity is then used to compute new pressure by solving a pressure Poisson equation. The second step is to solve for the new velocity also using the intermediate velocity and pressure. For temporal discretization, we use explicit time stepping.

A fluid region with a dimension of $30 \times 30 \times 40$ DPD units is surrounded by a solid wall consisted of fixed particles. At a particle density of 8, we set the DPD parameter $\alpha = 9.375$ and $\gamma = 13.5$ to counter the effect of compressibility due to the fast moving wall at a velocity of 1.833. This set of parameters results in a reference Reynolds number $\mathrm{Re}_{\mathrm{ref}} = 200$. To create the no-slip cavity lid and wall, we used an effective dissipative coefficient for liquid–solid interactions [78]. For the cavity flow, we used the same projection method as in the Poiseuille flow example, with



(a) (b)

Figure 4.10: (a) Rate of convergence of a two-dimensional cavity flow with `SPASD`. The normalized error ϵ_{l_2} at T_{final} is plotted for two different estimated Reynolds number. (b) The corresponding termination condition C_{TC} is calculated according to Eq. (4.1). Solid navy lines represent the iterations before the error threshold, and dotted yellow lines after the error threshold.

the consideration that the velocity profile consists of two components. As a result, both x and y components are recorded for each grid node. Because the grid in the coarse propagator spans in two dimensions, the mapping procedure for this example is slightly modified from that of plane-flow. The velocity of a particle is the linear interpolation of two nearest grid values. We used the same filtering techniques as in the prior examples.

Because an analytical solution does not exist for cavity flow, a reference solution is constructed by averaging the velocity fields from fifty independent serial DPD simulations. For the purpose of analysis, instead of particle velocity fields we inspect the equivalent stream function. The stream function can be easily obtained by solving the Poisson equation $-\nabla^2\psi = (\nabla \times \boldsymbol{u}) \cdot \hat{z}$. We then compute the normalized l_2 error ϵ_{l_2} between simulated and reference stream functions.

To test `SPASD`'s flexibility, we falsely assume that the fluid is much less viscous in the macroscopic model with a Reynolds number $\text{Re}_{\text{est}} = 600$ which is $3\times$ the reference Reynolds number Re_{ref}. It is important to note that the falsely assumed

viscosity would lead to an incorrect vortex location and, consequently, vastly different velocity field. As a result, the vortex from our SPASD simulation does not align with the reference vortex in early iterations shown in Fig 4.9. The mis-alignment is then iteratively corrected. We again calculate ϵ_{l_2} at T_{final} and plot it against iterations in Fig. 4.10(a). The rate of convergence is no longer exponential as it was for the Poiseuille plane-flow. This is due to the notion that in two-dimensional space the l_2 error might not decay in a shortest path. For this particular case, it took 5 iterations to reach the threshold error $\epsilon_{\text{Threshold}} = 1\%$ as shown in Fig. 4.10(a). We also repeated the experiment with an estimated Reynolds number Re_{est} that is equal to the reference Re_{ref}. As expected, it required fewer iterations to converge. The termination criterion was also computed and plotted in Fig. 4.10(b). It exhibits the same shortcoming as in the Poiseuille flow - no clear separation between iterations above and below $\epsilon_{\text{Threshold}}$. Further discussion on this issue is presented in Section 4.4.

4.4 Summary & Discussion

We have developed a supervised parallel-in-time algorithm for stochastic dynamics (SPASD). As an extension to the Parareal algorithm, SPASD uses heterogeneous solvers, e.g., a Navier-Stokes solver as a predictor and the DPD method as a corrector, to accelerate stochastic particle simulations by decomposing the temporal domain. The predictor, also known as the coarse propagator, solves the macroscopic model, which is generally in the form of partial differential equations, in serial. Because the microscopic model converges to continuum macroscopic models in the scale limit, the time evolution of the macroscopic system then serves as an initial condition for all time subdomains. The corrector, also known as the fine propagator, recovers the micro-dynamics with expensive stochastic simulations in parallel. While here

we demonstrated SPASD for hydrodynamics, the same framework can be applied to other problems, where long-time integration is hampered by the small characteristic time scale of the micro-dynamics.

To summarize, we first presented the SPASD algorithm and analyzed its theoretical speedup. We then discussed the importance of filtering and demonstrated its effect with an example problem. The accuracy and convergence of SPASD was subsequently tested. We showed that the transient and final solutions match the analytical/reference solutions even when our estimations to the system's mean-field behavior is far from the true behavior. More importantly, the SPASD algorithm is able to preserve the stochastic fluctuations of the microscopic model. Lastly, we demonstrated that SPASD provides better parallel efficiency and thus better scalability than the conventional domain decomposition method for long-time simulations.

There are three comments and suggestions we would like to make regarding SPASD. First, although the solution converges regardless the value of estimated viscosity, an overly rough estimation requires a large number of iterations. As for the Poiseuille flow example, a rough estimate at $10\times$ true viscosity requires 20 iterations, whereas only 5 iterations are needed for a better estimate at $2\times$ the true viscosity. In light of this correlation, we strongly encourage using an informed estimate of the macroscopic parameters in order to maximize the efficiency. Second, if the data from one microscopic state is insufficient to determine a mean-field quantity, the projection operation can be performed with data from multiple states with temporal-proximity. With the additional data, the mean-field quantity can be determined with higher precision. Lastly, we would like to discuss the issue of the termination condition. Given that SPASD is a general parallel-in-time algorithm, the termination condition presented in Section 4.2.2 should be application-dependent. The stochastic noise is embedded in the refined solution. $C_{\text{Tolerance}}$ thus should be adjusted according to

the magnitude of the noise. In the examples, we chose the normalized l_2 error as the quantity of interest C_{TC}. The termination criterion is satisfied when C_{TC} falls below a threshold (see Eq. (4.1)). For the Poiseuille flow, we computed C_{TC} for all iterations as shown in Fig. 4.6(b). When the parameter estimation is accurate as in the case of $2\times$ reference viscosity, C_{TC} fluctuates mildly as indicated by the range of confidence interval and drops sharply to a level that satisfies the termination condition. However, when estimation is rough, the confidence interval is relatively large, which might cause the simulation to stop prematurely. In this case, we recommend using noise-filtered solution in the error calculation. Another option is to define a different quantity of interest as the termination criterion with a redefined termination condition. Finally, it is important to mention that the use of filtering helps to damp the high-frequency noises in the mean-field quantities, which only acts as a rough predictor for the fine stochastic solver. Regardless of the choice of filtering algorithms, the final solution obtained with `SPASD` contains all the statistics of the stochastic fluctuations and their correlations, consistent with unsupervised (serial in time) stochastic dynamics.

Chapter Five

Supervised parallel-in-time algorithm for long-time Lagrangian simulations of stochastic dynamics: Application to blood flow in zebrafish

5.1 Introduction

The efficiency of the vascular network rests on the balance of two opposing requirements on network structures [112]. The demand for oxygen and nutrients necessitate a dense network, such that tissue cells are within the oxygen diffusion distance. A dense mesh-like network meets the requirement, but at the expense of high resistance to flow. Conversely, a tree-like network can minimize flow resistance but lacks uniform spatial coverage [112]. The actual vascular networks adapt a combination of mesh-like and tree-like networks to supply oxygen with approximately uniform density. Furthermore, it can remodel the existing network to adapt to new demands. Thus, vascular network formation is a complex problem of biological optimization [112] in disguise.

Angiogenesis and pruning, along with vessel formation (vasculogenesis), are the main processes involved in vasculature formation. They are essential for biological functions including growth, wound healing and aging [132]. Angiogenesis is the process of new blood vessel formation from existing vascular structure [106]. Vascular networks formed by sprouting angiogenesis subsequently undergo remodeling (*e.g.* vascular pruning and regression of particular vascular branches) to form a mature and efficient vasculature [106].

Aside from the desire to reveal nature's design, a deep and thorough understanding of those processes also provide therapeutic value. As an indication of cancer and various ischaemic and inflammatory diseases, pathological angiogenesis provides new blood vessels needed to sustain abnormal growth, *e.g.* tumors [16]. Vessel regression is an emerging field of research because of its therapeutic implications such as tumor inhibition. Further improvements will require a better understanding of angiogenesis

and pruning.

Both angiogenesis and pruning were formerly understood as chemical driven processes. However, recent studies suggest that they are both mechanically and chemically driven [106, 7, 51]. Shear stress on vessel wall stimulates vessel regression through chemical signaling [36, 105], and is induced by red blood cell movements [127, 47, 94]. A mechanically accurate model of blood with discrete red blood cells can accurately capture the effects of microcirculation on angiogenesis and pruning.

As an emerging model organism, zebrafish has many advantages over the traditional mouse model. Most notably, most vertebrate tissue of zebrafish larvae are transparent, which simplifies the examination of the vasculature *in vivo*. Because of this, observations of angiogenesis and pruning, and the subsequent hypothesis proposal and testing are easily performed [18]. Furthermore, research has shown mutual translatability of findings between zebrafish and human vascular biology [19, 79]. Thus, zebrafish as a tool to study human vasculature has been embraced by the community.

Although discrete models of blood can capture complicated interactions between blood components and the vessel wall, it is costly to simulate such systems in practicality. An adult female zebrafish has a body length of 3.87 centimeters and an average blood volume of 17 microliters [133], which translates to $5.134E7$ red blood cells [90]. A naive simulation (i.e. serial in time and space) with discrete red blood cells is unfeasible even for a sub-system that is 1% of the volume. A typical strategy to accelerate a large-scale simulation is spatial decomposition, *i.e.*, decompose the domain in space and then simulate all sub-domains concurrently with additional computing resources. This approach works very well for Lagrangian methods as they are highly parallelizable with good strong and weak scalings [11, 101]. As a result, it has

been incorporated into many well-known particle simulators [101, 17, 103, 80, 117].

Besides the massive spatial coverage, a lengthy physical time scale is another bottleneck for Lagrangian particle simulations. Since the atomistic details are explicitly modeled, the maximum time step is limited by the smallest oscillation period of the fastest atomic motion, typically on the order of several femtoseconds [31]. Consequently, the physical time scales accessible to those simulations are too short to address interesting long-time dynamics. In the example of thrombus formation caused by platelet margination and adhesion, the characteristic time for platelet activation is a few microseconds, and the formation processes usually require at least several hours to days [57, 111]. With a time step on the order of one microsecond, a clinically meaningful result for one-day time integration require 10^{11} time steps with the explicit platelet models [33], deeming it computationally prohibitive. To break the bottleneck, a supervised parallel-in-time algorithm for long-time Lagrangian simulations of stochastic dynamics (`SPASD`) was proposed by Blumers et al [11]. Inspired by parareal [82], `SPASD` is a predictor-corrector type algorithm for which the high-fidelity Lagrangian simulator (aka fine solver) is supervised by a low-fidelity continuum model (aka coarse solver). The results of the Lagrangian simulation then correct the mean-field prediction and provide the proper microscopic details (e.g., consistent fluctuations, correlations, etc.).

In this work, I attempt to accelerate massive Lagrangian simulations of blood flow in zebrafish. In addition, Spatial Decomposition (SD) method is applied in conjunction to additional speedup. The reminder of this paper is organized as follows: I first introduce `SPASD` for the application of blood flow in section 5.2.1 and 5.2.2. I then explain in detail the coupling scheme in our implementation in section 5.2.3. Additionally, the section contains the software architecture that enables real-time data transfer in massive parallel simulations. In section 5.3, I present quantitative

results of simulations via our SPASD framework. Finally, section 5.4 contains a brief summary and lengthy discussion.

Contributions: The one-dimensional model is set up and calibrated by my collaborator, Minglang Yin. Prof. Yosuke Hasegawa from the university of Tokyo digitized the scanned images from experiments. He also extracted the centerline from the 3-dimensional reconstruction of the vascular network.

5.2 Algorithm

5.2.1 Methods

For this application (Table 5.1) of SPASD, the high-fidelity Lagrangian simulator (aka fine solver) is Dissipative Particle Dynamics (DPD). DPD is commonly used for simulating fluid. Each DPD particle is represented explicitly by its position and velocity. This offers a bottom-up approach that resolves motion with Newton's equation and provides a detailed representation of various types of fluid. The mixture of single DPD particles and mesh-based red blood cells has been used to model blood flow in vessels. The high-fidelity DPD is complemented by the low-dimensional fluid model that is solved with Discontinuous Galerkin (DG). The reduced-order DG can provide a rapid estimation of the blood flow, which will be fed into the more expensive DPD.

The images of zebrafish are provided by our collaborator Dr. Hiroyuki Nakajima at the National Cerebral and Cardiovascular Center in Japan. To reconstruct the realistic 3D vasculature, a live zebrafish was scanned under a confocal microscope,

	Coarse Solver (CS)	**Fine Solver (FS)**
Name	NEKTAR-1D	USER MESO 2.0
Type	Continuum solver	Particle solver
Method	Discontinuous Galerkin	Dissipative Particle Dynamics (DPD)
Dimensionality	One-dimension	Three-dimension
Code	Serial	Parallel
Hardware support	CPU	GPU

Table 5.1: Features of the coarse solver and fine solver in SPASD.

resulting in a stack of 2D images. I stitched the resulting images and corrected confocal distortion, as seen in Figure 5.1. The digital replica was then used to build a particle system that is identical to the scanned images. Additionally, the centerlines of these vessels were extracted for NEKTAR-1D.

5.2.1.1 Dissipative Particle Dynamics

Each DPD particle is represented explicitly by its position and velocity. Its motion is governed by Newton's equations of motion [49]:

$$\frac{d\mathbf{r}_i}{dr} = v_i, \tag{5.1}$$

$$\frac{d\mathbf{v}_i}{dt} = \mathbf{F}_i = \sum_{i \neq j} \left(\mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R \right), \tag{5.2}$$

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 $\mathbf{F}_{ij}^C$, dissipative force $\mathbf{F}_{ij}^D$, and corresponding random force $\mathbf{F}_{ij}^R$ from particle j within a radial cutoff r_c of particle

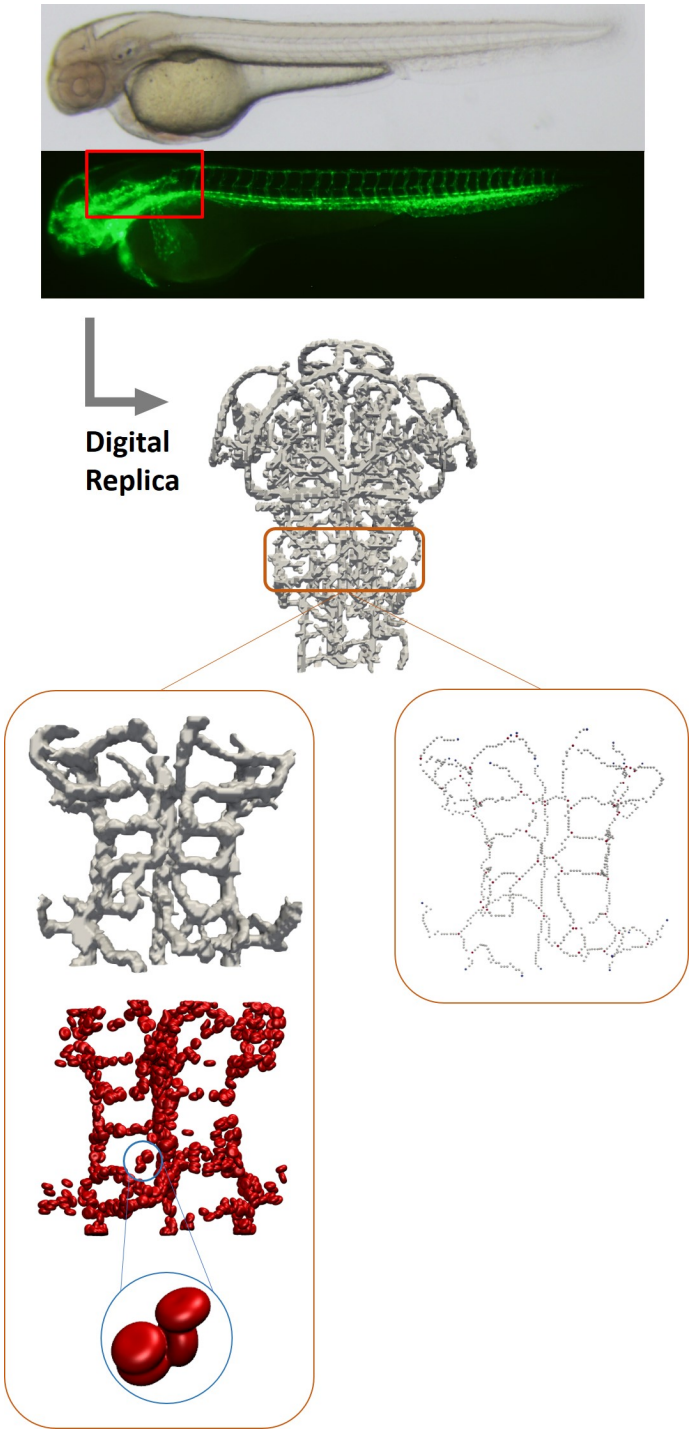


Figure 5.1: Geometry reconstruction for the high- and low-dimensional representations. I choose a zebrafish larva at approximately two days postfertilization (dpf) as our model. Specifically, I focus on the hindbrain of a living zebrafish. The images were obtained with confocal microscopy by our collaborators at the National Cerebral and Cardiovascular Center in Japan. I convert the scan into a digital replica, which is then used to construct a particle system that is identical to the real vasculature. Additionally, the centerline of the these vessels is extracted for the continuum solver.

i. They are expressed as [42]:

$$\mathbf{F}_{ij}^{C} = \alpha_{ij}\, \omega_C(r_{ij})\, \mathbf{e}_{ij}, \tag{5.3}$$

$$\mathbf{F}_{ij}^{D} = -\gamma_{ij}\, \omega_D(r_{ij})\, (\mathbf{e}_{ij} \cdot \mathbf{v}_{ij})\, \mathbf{e}_{ij}, \tag{5.4}$$

$$\mathbf{F}_{ij}^{R} = \sigma_{ij}\, \omega_R(r_{ij})\, \xi_{ij}\, \Delta t_f^{-1/2}\, \mathbf{e}_{ij}, \tag{5.5}$$

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. Δt_f is the time step of the fine propagator, and ξ is a symmetric Gaussian random variable with zero mean and unit variance [42]. The strengths of the conservative, dissipative, and random force are α_{ij}, γ_{ij} and σ_{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 [27]:

$$\sigma_{ij}^2 = 2\gamma_{ij} k_B T, \qquad \omega_D(r_{ij}) = \omega_R^2(r_{ij}), \tag{5.6}$$

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 $\omega_D(r) = \omega_R^2(r) = (1 - r/r_c)^2$ 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.

5.2.1.2 Blood model

The particle-based blood flow model is based on the dissipative particle dynamics approach [99, 29]. 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 Section 5.2.1.1, 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 by [99, 29]

$$V_{WLC} = \frac{k_B T h_m}{4p} \frac{\frac{h}{h_m}^2 (3 - 2\frac{h}{h_m})}{1 - \frac{h}{h_m}}, \tag{5.7}$$

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

$$V_{POW} = \begin{cases} \frac{k_p}{(m-1)h^{m-1}}, & m \neq 1, \\ -k_p \log(h), & m = 1, \end{cases} \tag{5.8}$$

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

$$\mathbf{F}_{ij}^D = -\gamma^T \mathbf{v}_{ij} - \gamma^C (\mathbf{v}_{ij} \cdot \mathbf{e}_{ij}) \mathbf{e}_{ij}, \tag{5.9}$$

$$\mathbf{F}_{ij}^R = \sqrt{2k_B T} \Big(\sqrt{2\gamma^T} (d\mathbf{W}_{ij}^S - tr[d\mathbf{W}_{ij}^S]\mathbf{I}/3) + \frac{\sqrt{3\gamma^C - \gamma^T}}{3} tr[d\mathbf{W}_{ij}]\mathbf{I} \Big), \tag{5.10}$$

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 $d\mathbf{W}_{ij}$.

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

by

$$V_{area} = \frac{k_g(A^t - A_0^t)^2}{2A_0^t} + \sum_j \frac{k_l(A_j - A_{0,j})^2}{2A_{0,j}}, \tag{5.11}$$

$$V_{volume} = \frac{k_v(V^t - V_0^t)^2}{2V_0^t}, \tag{5.12}$$

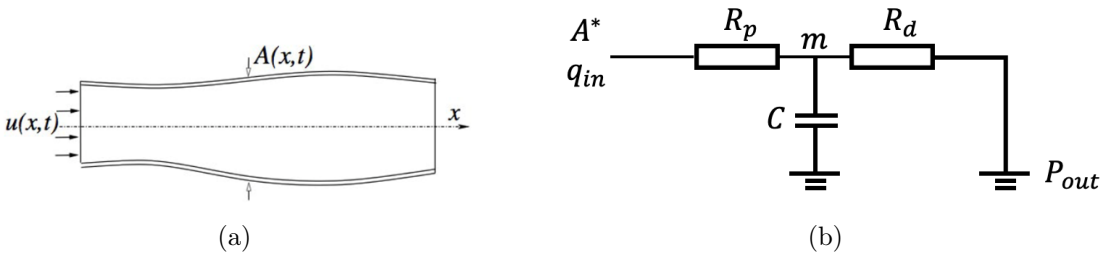
where j is the triangle index. k_g, k_l, and k_v are global area, local area, and global volume constraints coefficients, respectively. The instantaneous area and volume of a RBC are denoted by A^t and V^t, whereas A_0^t and V_0^t represent the equilibrium area and volume. A_0^t is calculated by summing 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

$$V_{bending} = \sum_j k_b\Big(1 - \cos(\theta_j - \theta_0)\Big), \tag{5.13}$$

where k_b is the bending constant, and θ_j is the angle between two neighboring triangles with the common edge j. I refer interested readers to references [99, 29] for more details.

5.2.1.3 1D solver

Under the assumptions of axial symmetry and radial displacement of the vessel wall (Figure 5.2a) [113][1], the modeling of blood flow in a compliant vessel can be reduced from full-order Navier-Stokes equations to an one-dimensional (1D) model. The reduced-order model has been widely used for blood flow simulations at a moderate [114] and low [97] 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.



(a) (b)

Figure 5.2: (a) Blood vessel as an symmetric tube. The assumptions of axial symmetry and radial displacements of the vessel wall reduce the full-order Navier-Stokes equations to an one-dimensional model. This figure is originally illustrated in [114]. (b) Schematic of the RCR boundary condition at outlets for the 1D model.

The 1D model builds on mass conservation and momentum equations [114]:

$$\frac{\partial A}{\partial t} + \frac{\partial AU}{\partial x} = 0 \tag{5.14}$$

$$\frac{\partial U}{\partial t} + \frac{1}{2}\frac{\partial U^2}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} = -\frac{1}{\rho}(K_r\frac{U}{A} - S), \tag{5.15}$$

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_r is the friction loss coefficient representing the viscous resistance of flow per unit length along the vessel. It can be expressed as $K_r = \frac{2\alpha\mu\pi}{\alpha-1}$, where $\alpha = 4/3$ assumes a laminar blood flow with parabolic velocity profile. I assume no tapering effect such as stenosis or atherosclerosis by setting $S = 0$.

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

$$p = p_{ext} + \beta\big(\sqrt{A} - \sqrt{A_o}\big), \tag{5.16}$$

$$\beta = \frac{\sqrt{\pi}hE}{(1-\nu^2)A_o}, \tag{5.17}$$

The pressure p is the summation of the external pressure p_{ext} and pressure variation, which is characterized by arterial wall compliance β and square root of area

difference. The Poisson ratio ν is taken to be 0.5, and h and A_o are the vessel wall thickness and reference cross-sectional area. E is the Young's modulus of the vessel wall.

I use the second-order Adams-Bashforth scheme to discretize and compute the time integration. For spatial discretization, 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. I refer interested readers to [113] for details.

The imposed boundary condition must satisfy conservation of mass, which constrains the flow rate at outlets. For the DPD model, I can impose Dirichlet type boundary conditions at inlets and outlets. However, for the 1D system, this would lead to a defective boundary condition [35] and a problem with well-posedness [104]. An alternative to the defective boundary condition is the Windkessel model[126], which is analogous to open-loop circuits. This approach captures the effective resistance (R) and compliance (C) of the neglected downstream vessels. In particular, I apply a three element "Windkessel" RCR boundary conditions to each of the outlets, as shown in Fig. 5.2b. The RCR boundary conditions are given as:

$$q_{in} = \frac{P(A^*) - P_m}{R_p} = C\frac{dP_c}{dt} + \frac{P_{out} - P_{A^*}}{R_d} \tag{5.18}$$

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

5.2.2 Supervised Parallel-in-time Algorithm for Stochastic Dynamics (SPASD)

Although the coarse and fine solvers (CS and FS) are DPD and DG respectively in this application, SPASD (Algorithm 4) can be used with any combination of a stochastic particle-solver and its continuum counterpart [11]. This flexibility is warranted by the deliberate choice of the state-variable, which is the medium between micro- and macro-scales. An appropriate state-variable must be easy-to-extract, and it must describe the state of the system in a relevant way. In our temporal multi-grid approach, I take axial volume-flux as the state-variable between CS and FS. The underlying reason is twofold. First, volume-flux is a state-variable in the 1D model. Its evolution, along with area, describes the state of blood flow over time. Second, matching volume-flux can preserve the flow profile in vessels, and the flow profile is often used to characterize blood flow. In the particle model, the volume-flux at a cross-sectional plane is calculated from the velocity of adjacent particles. It can be easily tuned by scaling up or down the component in the centerline direction while affixing the other components. Beside axial volume-flux, axial velocity is the other state variable in the one-dimensional model. However, initializing the three-dimensional DPD system from the one-dimensional axial velocity would not preserve the flow profile.

The governing equations of the macroscopic model are vastly different from those of the microscopic model. Consequently the solution representing the macroscopic state is vastly different from the one representing the microscopic state. The solution from CS can be mapped to a microscopic state via a mapping operation (denoted by mapping operator $\mathscr{R}$). Conversely, the solution from FS can be obtained from a microscopic state via a projection operation (denoted by projection operator $\mathscr{P}$).

Algorithm 4 `SPASD`

1: Initialization:

Compute initial iteration with CS : $Q_0^{n+1} = \mathcal{G}(Q_0^0)$ for all $0 \leq n \leq N$.

2: Assume sequence $\{Q_k^n\}$ is known for some $0 \leq n \leq N$ and $k \geq 0$:

Correction:

(a) Filter solution to remove noise: $\mathscr{F}\{Q_k^n\}$

(b) Advance with CS in serial: $\mathcal{G}(\mathscr{F}\{Q_k^n\})$ for all $0 \leq n \leq N-1$.

(c) Map the macroscopic state to microscopic state: $\mathscr{R}\{Q_k^n\}$.

Skip mapping if $Q_k^n - Q_{k-1}^n < \Delta_{\text{th}}$ where Δ_{th} a user-defined threshold.

(d) Advance with FS in parallel: $\mathcal{F}(\mathscr{R}\{Q_k^n\})$ for all $0 \leq n \leq N-1$.

(e) Project the microscopic state to macroscopic state: $\mathscr{P}\{\mathcal{F}(\mathscr{R}\{Q_k^n\})\}$

(f) Compute the correction: $\delta_k^n \equiv \mathscr{P}\{\mathcal{F}(\mathscr{R}\{Q_k^n\})\} - \mathcal{G}(\mathscr{F}\{Q_k^n\})$ for all $0 \leq n \leq N-1$.

Prediction:

Advance with CS in serial: $\mathcal{G}(\mathscr{F}\{Q_{k+1}^n\})$ for all $0 \leq n \leq N-1$

Refinement:

Combine the correction and the prediction terms:

$Q_{k+1}^{n+1} = \mathcal{G}(\mathscr{F}\{Q_{k+1}^n\}) + \mathscr{P}\{\mathcal{F}(\mathscr{R}\{Q_k^n\})\} - \mathcal{G}(\mathscr{F}\{Q_k^n\})$ for all $0 \leq n \leq N-1$.

3: Repeat *Step 2* to compute Q_{k+2}^n for all $1 \leq n \leq N$ until the solution converges.

Usually methods such as weighted kernel sampling and interpolation are employed for the mapping and projection operations. In the current application, the volume-flux is computed at the centerline points, which are shared by both models because the 1D geometry is reduced from the 3D geometry. Therefore, the mapping the projection operations are not required here.

Noise filtering, denoted by operator $\mathscr{F}$, is essential to the success of `SPASD` because the noise from the FS would accumulate with time. The accumulation will cause the refined solution to diverge from the true solution if left unattended. In this application, noise filtering is embedded in the coarse propagation. The governing equations of CS exhibit strong noise smoothing characteristics as the spatial derivative on volume-flux is only first order. Coupled with fixed boundary values, the system can effectively eliminate noise over time.

I customized SPASD for the application of blood flow in this publication. Algorithm 4 is identical to the original SPASD except one addition – skip mapping if $Q_k^n - Q_{k-1}^n < \Delta_{\text{th}}$ where Δ_{th} a user-defined threshold. This modification is intended to hedge against the non-uniform particle distribution due to the explicit modeling of RBCs. Since volume-flux is computed using particle density, non-uniform particle distribution makes the volume-flux slightly inaccurate, as demonstrated in the Y-bifurcation benchmark example. When the difference between two consecutive iterations $Q_k^n - Q_{k-1}^n$ is less than 10%, I found that skipping the mapping $\mathscr{R}\{Q_k^n\}$ avoids inaccurate re-initialization of time-subdomains.

5.2.3 Multiscale framework

Coupling two solvers at different scales are common in multiscale frameworks. However, it can be conceptually challenging for solvers of different classes. In this section, I will explain in detail the coupling scheme between DPD and DG. For clarity I will outline the interplay between the solvers in Section 5.2.3.2. Furthermore, I will describe the implemented software architecture that enables real-time data transfer in massive parallel simulations.

5.2.3.1 Flux computation

The 1D-model uses a relative linear coordinate system in conjunction with a binary tree. Because the low- and high-dimensional models describe the same geometry, the centerline of the 3D vascular network is the coordinate of the reduced 1D system. As shown in Section 5.2.1.3, volume-flux can be readily computed for each centerline point in CS.

On the other hand, the particle system can be down-scaled by decomposing the 3-dimensional space into cells. In this approach, the neighboring particles are lumped into one cell, and each cell is represented by its center referred to as the cellular node, as illustrated in Figure 5.3. By averaging the dynamical state of particles in a cell as a whole, volume-flux for each cell can be approximated as

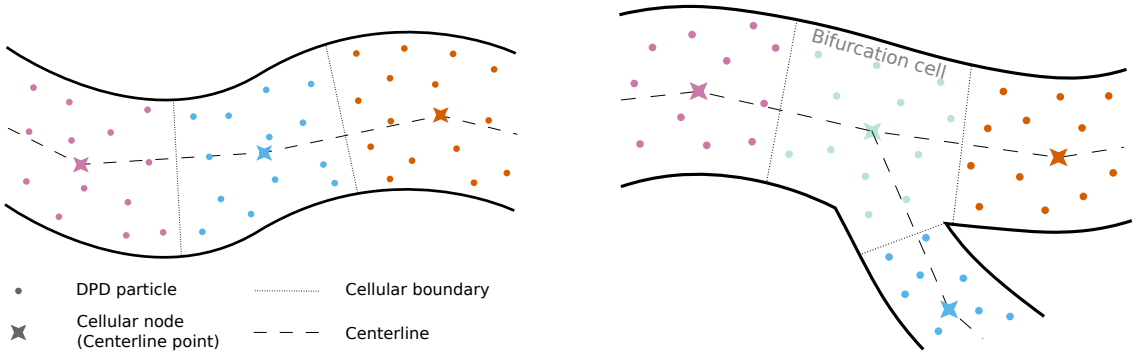
$$Q^D \approx \sum_i \frac{\mathbf{v}_i \cdot \hat{\mathbf{n}}}{h\rho}, \tag{5.19}$$

where $\mathbf{v}_i$ is the velocity of particle i, and $\hat{\mathbf{n}}$ is the direction of the centerline passing through the cell. Because volume-flux is the rate of flow across a plane, I need to divide by the cell thickness h and the particle density ρ assuming each particle has unity mass. The volume-flux at bifurcation cells are not computed since they are undefined.

Because the coordinate systems in CS and FS overlap, each element in CS corresponds to one cell in FS. The centerline connecting those cells runs through all cellular nodes. This overlap establishes connections between 1D and 3D geometries in CS and FS, respectively.

5.2.3.2 Coupling scheme

As outlined in Table 5.2, the coupling procedure starts with associating particles to their respective cells after a fine propagation. The volume-flux Q^D of each cell is then computed (as described in section 5.2.3.1) and pushed to CS. CS uses the fetched Q^D to compute an updated volume-flux Q^N. It subsequently pushes Q^N to the FS as the initial condition. Particle velocities are adjusted accordingly to match Q^N for each cell. This step is commonly referred to as the up-scaling in the context



(a) (b)

Figure 5.3: Particle-cell association. (a) In our approach, the neighboring particles are lumped into one cell, and each cell is represented by its center referred to as the cellular node. Volume-flux is computed on a cellular basis by averaging the dynamical state of particles the cell. (b) Because of the association scheme, the volume-flux at a bifurcation cell is undefined and thus not computed.

	Continuum solver		**DPD solver**
1			Run the DPD simulation.
2			Down-scaling: Associate particles to their respective cells; compute volume-flux $\{Q^{\mathrm{D}}\}$ for each cell by summing over contributions from enclosed particles
3	Fetch $\{Q\}$ on centerline points	◀	Push $\{Q^{\mathrm{D}}\}$ on cellular nodes
4	Update $\{Q\}$ with corrections as described in `SPASD`		
5	Push $\{Q\}$ on centerline points from the next timestamp	▶	Fetch $\{Q^{\mathrm{N}}\}$ on cellular nodes
6			Up-scaling: Tune particle velocity to match the corresponding cellular volume-flux $\{Q^{\mathrm{N}}\}$ using $\{Q^{\mathrm{D}}\}$.
	Repeat from 1		Repeat from 1

Table 5.2: Outline of the coupling scheme between the continuum and DPD systems.

of multiscale coupling.

Given the particle system can be in any dynamic state prior to the up-scaling step, I found that a universal mapping cannot effectively match the target volume-fluxes. Instead, I propose a mapping that is pre-conditioned on the current state:

$$\mathbf{v}_i^{\text{New}} \leftarrow \begin{cases} \frac{Q^N}{A}\hat{\mathbf{n}} & \text{, if } 1-\alpha < F < 1+\alpha \quad (5.20) \\ \mathbf{v}_i + (\mathbf{v}_i \cdot \hat{\mathbf{n}})(\frac{Q^N}{Q^D} - 1)\hat{\mathbf{n}} & \text{, otherwise} \quad (5.21) \end{cases}$$

A is the average area of a cell, which I approximate as the cross-sectional area at the cellular node. When the current volume-flux Q^D and the target volume-flux Q^N are on the same order of magnitude and non-zero, Q^N/Q^D is close to one, and the particle velocity is modified according to Equation (5.21). The goal is to increase the velocity in the direction of centerline such that it reflects the new flux Q^N. On the other hand, if Q^D is close to zero and Q^N is not, Q^N/Q^D becomes unreasonably large. This scenario typically occurs in the first iteration when the particle system (or part of it) was under initialization. In this case, Equation (5.21) cannot accurately tune particle velocities. To circumvent this issue, particle velocity is modified according to Equation (5.20) when $1-\alpha < F < 1+\alpha$, where $F \equiv (Q^N - Q^D)/Q^N$, where α is a pre-determined parameter between 0 and 1 regulating the method selection. For each combination of the signs of Q^N and Q^D, I tabulated the preferred method given the range of F in Table 5.3.

CS and FS have their respective coordinate systems: linear coordinate system for the low-dimensional model and full 3D coordinate system for high-dimensional model. In order to couple, there must be a directional mapping for translating vectorial quantities between the solvers. $\hat{\mathbf{n}}$ in Equation (5.19) and (5.20) takes on the role of directional mapping. Since the binary-tree geometry is constructed based

Case	Range			Method
$Q^N: - \longrightarrow +$ $Q^D: - \longrightarrow +$	$0 < Q^D < \alpha Q^N$	$\leftrightarrow$	$1 - \alpha < F < 1$	Equation (5.20)
	$\alpha Q^N < Q^D$	$\leftrightarrow$	$F < 1 - \alpha$	Equation (5.21)
$Q^N: - \longrightarrow +$ $Q^D: - \longleftarrow +$	$-\alpha Q^N < Q^D < 0$	$\leftrightarrow$	$1 < F < 1 + \alpha$	Equation (5.20)
	$Q^D < -\alpha Q^N$	$\leftrightarrow$	$1 + \alpha < F$	Equation (5.21)
$Q^N: - \longleftarrow +$ $Q^D: - \longleftarrow +$	$\alpha Q^N < Q^D < 0$	$\leftrightarrow$	$1 - \alpha < F < 1$	Equation (5.20)
	$Q^D < \alpha Q^N$	$\leftrightarrow$	$F < 1 - \alpha$	Equation (5.21)
$Q^N: - \longleftarrow +$ $Q^D: - \longrightarrow +$	$0 < Q^D < -\alpha Q^N$	$\leftrightarrow$	$1 < F < 1 + \alpha$	Equation (5.20)
	$-\alpha Q^N < Q^D$	$\leftrightarrow$	$1 + \alpha < F$	Equation (5.21)

Table 5.3: The mapping to target volume-flux in the up-scaling step is preconditioned on the current step. I tabulated the preferred method for a given combination of Q^N and Q^D.

on the 3D vascular network, I can compare the orientation of each branch. The orientation of a branch is defined as the direction away from the bifurcation point. If the orientation aligns with the direction of the coordinate system in the full 3D network, $\hat{\mathbf{n}}$ is positive for the corresponding axis. If the orientation is the opposite, $\hat{\mathbf{n}}$ is negative.

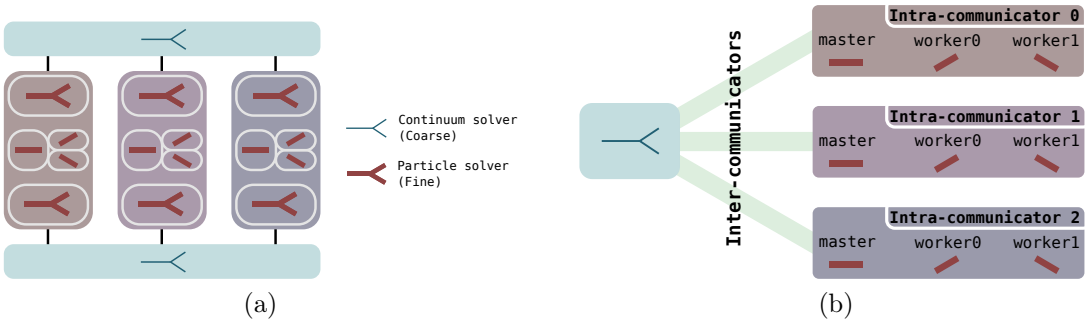
5.2.3.3 Parallel implementation

The continuum and particle solvers are, by themselves, stand-alone solvers with complex code structures. Injecting codes into the solvers for coupling, while complying with the structure of the software, is a challenging task. To minimize code refactoring, I use Multiscale Universal Interface (MUI). MUI is a library that facilitates the concurrent coupling of heterogeneous solvers [120]. 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.

I illustrate the information-exchange pipeline in Figure 5.4a. 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 Figure 5.4b. 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.



(a) (b)

Figure 5.4: Pipeline of the parallel implementation. (a) Coarse-solver (i.e. continuum solver) sends the volume-flux to the fine-solvers as initial conditions. The fine-solvers then propagate the systems concurrently. For each fine-propagator, its simulation domain is decomposed into multiple sub-domains for additional layer of parallelism. (b) The dual level of parallelism (i.e. spatial and temporal) is achieved owing to the clever arrangement of intra-solver and inter-solver MPI communicators.

5.3 Numerical Results

5.3.1 Benchmark problem – Y bifurcation

The Y-bifurcating channel is composed of one parent pipe and two identical daughter pipes, with a branch half-angle 26 degrees, as illustrated in Figure 5.5. In this problem, I prescribe volume-fluxes at the inflow and outflow boundaries by imposing a velocity profile in the boundary zones. The velocity profiles and shear-stress are then calculated by sampling particles in the sample zones. The sample zones are placed away from the bifurcating point and boundaries so that the flow profiles are steady in the sample zones and can be meaningfully compared.

Results obtained with our framework are compared against the reference results, which are obtained with identical, but serial, simulations. This allows us to directly assess the implications of our parallel-in-time framework. The deviation from the

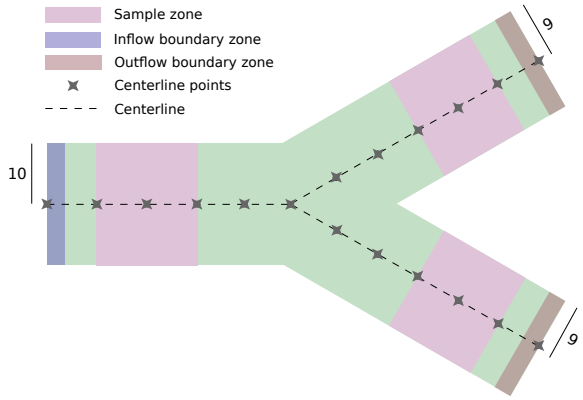


Figure 5.5: Central cut-plane of the 3-dimensional bifurcating channel. The channel is composed of one parent pipe and two identical daughter pipes. The inflow and outflow boundary conditions are enforced in the boundary zones. I sample statistics of the flow, such as velocity and flux, in sample zones.

reference solution is quantified using l^2 relative error norm ϵ, expressed as

$$\epsilon \equiv \frac{\left\|Q^{\mathrm{ref}} - Q^{\mathrm{sim}}\right\|_{l^2}}{\left\|Q^{\mathrm{ref}}\right\|_{l^2}} \equiv \frac{\left(\sum_{i=1}^{N}(Q_i^{\mathrm{ref}} - Q_i^{\mathrm{sim}})^2\right)^{1/2}}{\left(\sum_{i=1}^{N}(Q_i^{\mathrm{ref}})^2\right)^{1/2}}, \tag{5.22}$$

where Q^{ref} and Q^{sim} are the reference and simulated results, respectively. By normalizing against the reference solution, ϵ represents the deviation.

To discount thermal noise, I average the results of 40 independent simulations. The averaged results are then compared against those of the reference. Although the stochastic noise is still present post-averaging, it is reduced to a manageable magnitude so that the validity of `SPASD` can be meaningfully inspected.

5.3.1.1 Simple fluid

To drive the time-dependent flow, I specify the volume-flux as a sinusoidal function of time at the inlet and outlet boundaries. For this simple fluid example, the volume-flux at the boundaries are prescribed as shown in Figure 5.6. To be more precise,

velocity of particles in the boundary zones is adjusted to match the desired flux. For the pipe flow, I use a parabolic velocity profile defined as $u(r) = c(R^2 - r^2)$ where r denotes radial distance, and c is the strength. In this example, c_{inflow} and c_{outflow} are 0.02 and 0.0152, respectively.

As illustrated in Figure 5.6, the temporal domain is divided into four sub-domains, each of which is associated with an independent DPD system. Because the temporal sub-domains of the four DPD systems do not overlap, the four systems can evolve in time concurrently. That is, the systems are initialized at $t = \{0,\ T/4,\ T/2,\ 3T/4\}$, and propagate to $t = \{T/4,\ T/2,\ 3T/4,\ T\}$ respectively. At those timestamps, I plot the refined volume-flux in Figure 5.7, and calculate l^2 relative error norm ϵ defined in Equation (5.22), which are tabulated in Table 5.4. I found that ϵ are less than 1% for all sub-domains, indicating good agreement between the reference and `SPASD`. Additionally, I found ϵ are in the same order across iterations, signifying convergence.

Additionally, I compared the velocity profiles and shear rates, i.e. the derivative of velocity profiles in the radial direction, shown in Figure 5.8 for further verification. Shear rates are obtained by first fitting the velocity profile to a parabolic equation, and then taking an analytic derivative. Again, I found the reference and `SPASD` solutions matches well for all iterations. Most importantly, I observe that the results and errors stay approximately the same regardless of the number of iterations. This signifies that one iteration is sufficient for convergence with our implementation.

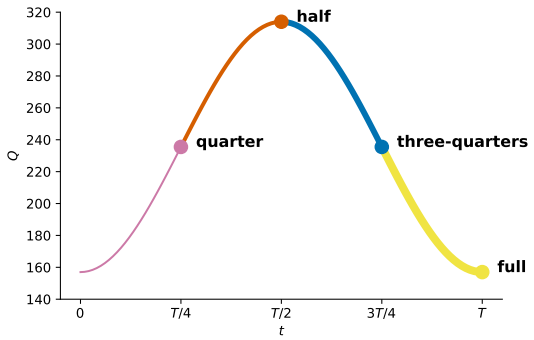


Figure 5.6: Time-dependent boundary condition for the simple fluid example. The volume-fluxes (Q) at the boundaries are prescribed as a sinusoidal function of time. The peak Q for the parent pipe is 314 as shown here. Since each daughter pipes has a volume-flux half of that of the parent pipe, its peak Q is 175. The temporal space, a full sinusoidal period, is divided into four quarters. The end of each quarter is marked on the plot.

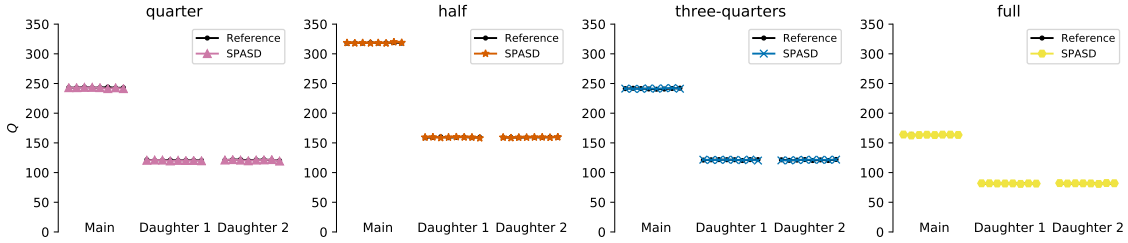
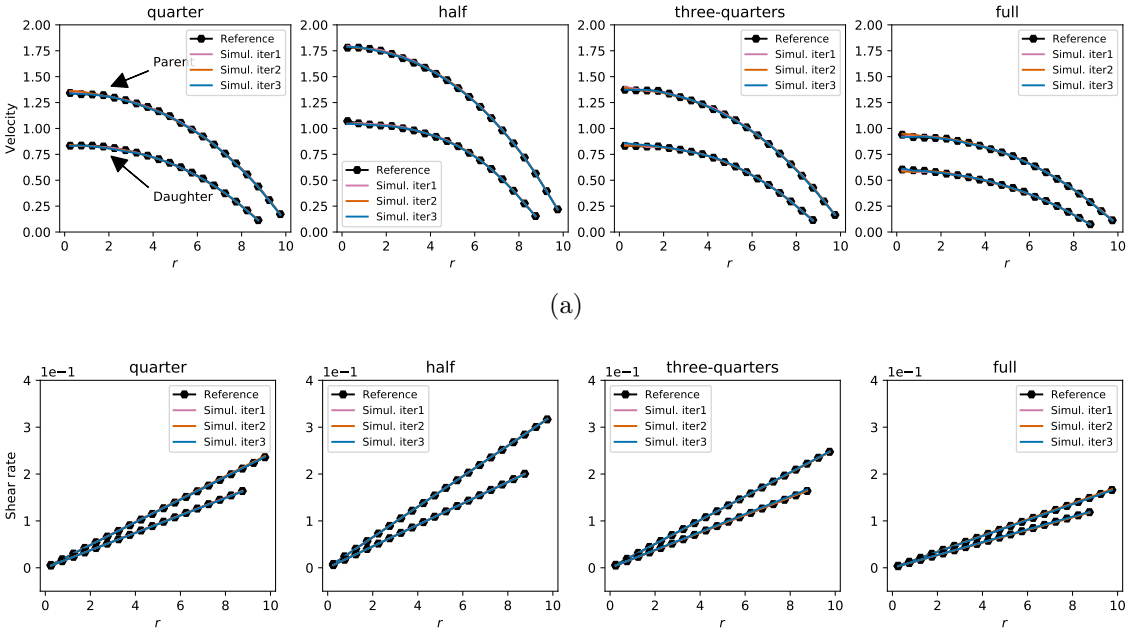


Figure 5.7: Volume-flux at specific times marked in Figure 5.5 in the simple fluid example. The refined volume-flux in each pipe is compared against the reference flux obtained via serial simulations. I see that `SPASD` is able to produce solutions very close to the reference.

	quarter	**half**	**three-quarters**	**full**
	Iteration 1	0.50%	0.51%	0.70%	0.70%
ϵ	Iteration 2	0.58%	0.37%	0.40%	0.75%
	Iteration 3	0.55%	0.36%	0.44%	1.01%

Table 5.4: l^2 relative error norm of the volume-flux defined in Equation (5.22), for the simple fluid example. ϵ less than 1% for all sub-domains indicates good agreement between the reference and `SPASD`.



(a)

(b)

Figure 5.8: (a) Velocity profiles and (b) shear rates, i.e. the derivative of velocity, at specific times marked in Figure 5.5, for the simple fluid example. I 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.

5.3.1.2 Complex fluid

The flow of complex fluid is driven similarly to the simple fluid – volume-flux at the boundaries are prescribed as a sinusoidal function of time as illustrated in Figure 5.6. The peak volume-flux at the parent and each daughter boundary is approximately 63 and 31.5, respectively. Because of complex fluid's non-Newtonian characteristics, a velocity profile with an exponential form $u(r) = ae^{br} + c$ is imposed at the boundaries to obtain the desired flux. Volume-flux can be adjusted by changing a and c in this form. In this example, the blood is a mixture of single particles and discrete RBCs at a hematocrit of 60%.

With the temporal domain decomposed similarly to the simple fluid's, I again compare volume-fluxes, velocity profiles and shear rates to those of the reference in Figures 5.9, 5.10a and 5.10b. Since shear stress at the wall is of particular interest owing to its effects on chemical activation, I also compute shear stress τ by binning the particles radially and averaging in bins. The shear stresses from 50 independent runs are plotted in Figure 5.10c, in which the error bar denotes one standard deviation from the mean. Because all bins have the same thickness, less data are available in bins toward the center, and the error bars are thus longer. As r increases, $\tau_{\text{reference}}$ matches τ_{SPASD} much better. I also plotted the refined volume-flux in Figure 5.9, and tabulated l^2 relative error norm ϵ in Table 5.5. Due to the uniformity of particle distribution in the complex fluids, the volume-flux and velocity profiles match those of the reference less precisely. I will discuss and expand on this topic further in Section 5.4.

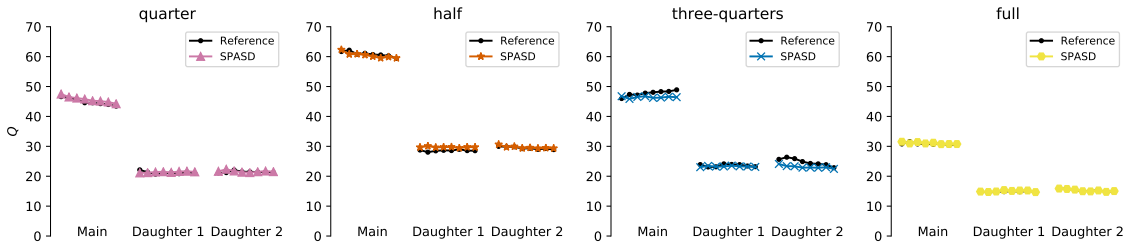
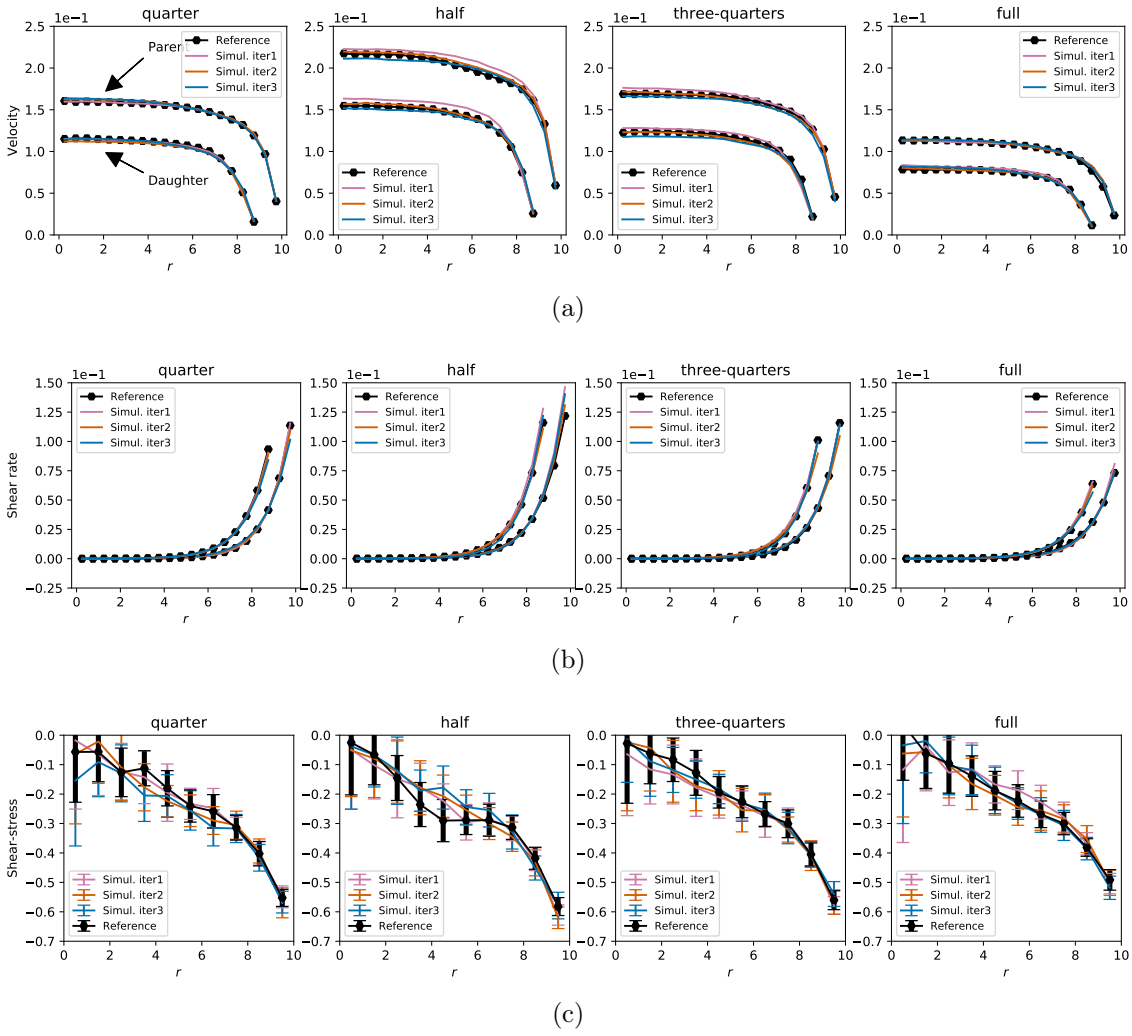


Figure 5.9: Volume-flux at specific times marked in Figure 5.6, for the complex fluid example. Due to the uniformity of particle distribution in the complex fluids, the volume-flux and velocity profiles match those of the reference less precisely. I will discuss and expand on this topic further in Section 5.4.



(a)

(b)

(c)

Figure 5.10: (a) Velocity profiles and (b) shear rates and (c) shear stress at specific times marked in Figure 5.6, for the complex fluid example. I 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.

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

Table 5.5: l^2 relative error norm of the volume-flux defined in Equation (5.22), for the complex fluid example.

5.3.2 Realistic vascularture – zebrafish hindbrain

5.3.2.1 Simulation Setup

I chose the zebrafish hindbrain (Figure 5.1) to test our method because of its complex geometry. 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 dpf developmental stage, the vessel diameters range from 10 to 20 micrometers, and the region is merely 224x197x160 micrometers in size. The compactness of the region and the complexity of the vascular network make it challenging for 1D solver to model it accurately, but it is an ideal benchmark for our method.

For computational stability, the 1D and 3D systems use their respective reduced units, which can be mapped to physical units. For the particle solver, the length scale is chosen to be $[L^{\mathrm{DPD}}] = 1 \times 10^{-6}$m. By the matching the viscosity of real blood to that of the DPD fluid, the time scale is calculated to be $[T^{\mathrm{DPD}}] = 1.2225 \times 10^{-3}$ s. On the continuum solver side, the length and time scales are chosen so that the volume-flux matches that of the real blood in zebrafish. In this example, I used $[L^{\mathrm{DG}}] = 1 \times 10^{-2}$ and $[T^{\mathrm{DG}}] = 1$ for stability and accuracy reasons. The details on mapping to physical units can be found in Appendix 5.5.2.

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 3554 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% [62]. To form a closed system with periodicity, 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, I impose a no-flux boundary condition on the other four sides (i.e. top, bottom, front and back).

5.3.2.2 Boundary & Initial Conditions

I again divide the temporal domain into four sub-domains, as illustrated in Figure 5.11. For the first iteration, *FS 1, 2, 3* and *4* propagates the systems concurrently, one for each sub-domain. At the end of 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.

Because the 1D solver relies on the centerline of the 3D particle system, the locations of the boundaries are the same for both solvers, illustrated in Figure 5.12. As expected, I prescribe the same set of fluxes as boundary conditions in both systems. To mimic the variation in blood flow from heartbeats, the flux at the inlets and outlets are prescribed as a periodic time-dependent function. Not knowing the exact rhythm at hindbrain, I simulate the rhythm of human heart for demonstration purposes, shown in Figure 5.13.

Our collaborators were able to measure the speed of RBCs in vessels by tracking

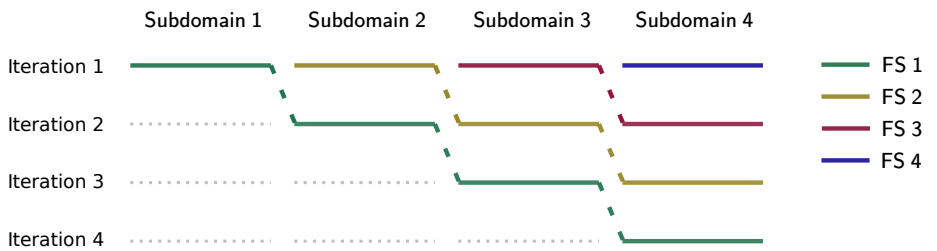


Figure 5.11: In the zebrafish hindbrain example, I again divide the temporal domain into four sub-domains. For the first iteration, *FS 1, 2, 3* and *4* propagates the systems concurrently, one for each sub-domain. At the end of first iteration, the solution in sub-domain 1 is accurate and does not need 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.

RBCs over time. Based on those statistics, I know the speed and direction of blood flow near inlets and outlets. For vessels too small for cells to fit in, the flow is assumed to be 100 μm/sec. Furthermore, I notice that the spread in measured speed is approximately 1.3 times the mean. With that distribution, the imposed cardiac rhythm is re-normalized so that the peak and trough match the maximum and minimum speed respectively for each inlets and outlets. In addition, I assume that the pulsatile flow is in-phase at inlets and outlets. Although this is not realistic, the phase difference in vessels of this size is very small and difficult to measure, according to experimental observations. So I make this assumption to simplify the implementation.

At 2 dpf, the heart rate is approximately 175 beats per minute [60], which is equivalent to 0.3429 second per beat. I split the time domain in half for this demonstration, and the two sub-domains are run in parallel with the particle solver.

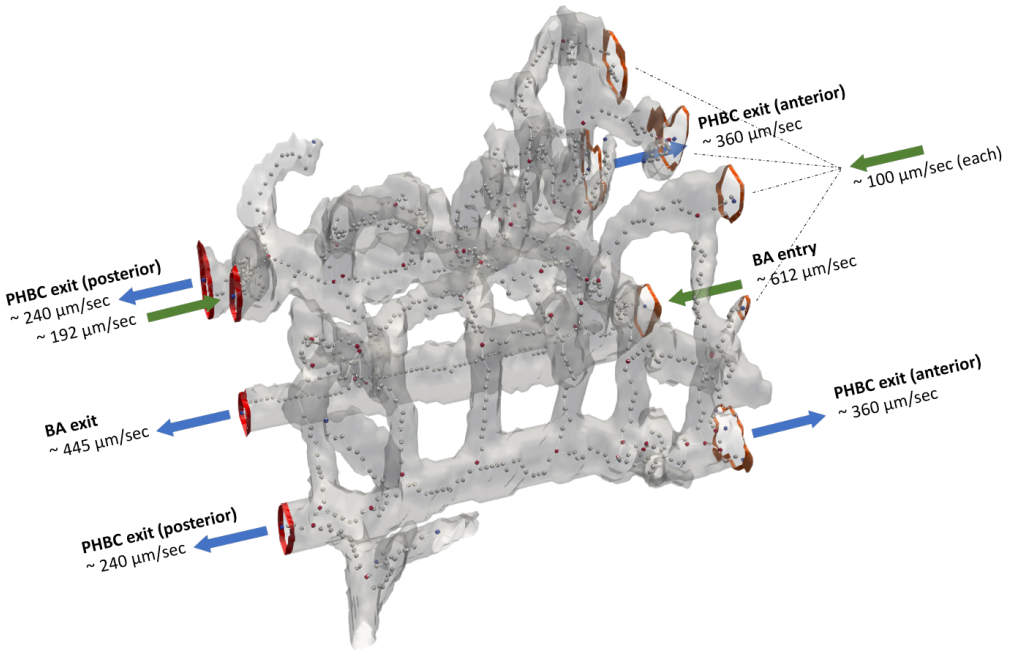


Figure 5.12: Overlap of the realistic 3D geometry and the 1D centerline. The blue nodes represent boundaries of the volume, and the red nodes represent bifurcations. The boundaries are highlighted in red, and the flow direction is marked with arrows. To mimic the variation in blood flow from heartbeats, the flux at the inlets and outlets are prescribed as a periodic time-dependent function shown in Figure 5.13.

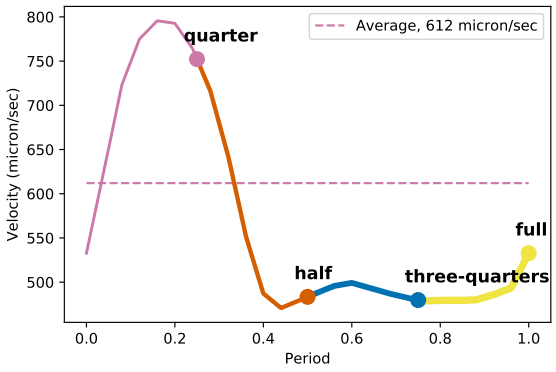


Figure 5.13: Not knowing the exact rhythm at hindbrain, I assumed the above cardiac rhythm for demonstration purposes. From analyzing experimental images, I extrapolate the distribution of RBC speed in each inlet and outlet. The rhythm is then re-normalized so that the peak and trough match the maximum and minimum speed for each boundary. For example, an inlet with an average speed of 612 μm/sec has a maximum velocity of 795.6 μm/sec (peak) and a minimum of 470.7 μm/sec (trough). The temporal space of a full cycle is divided into four quarters, with the end of each quarter marked on the plot.

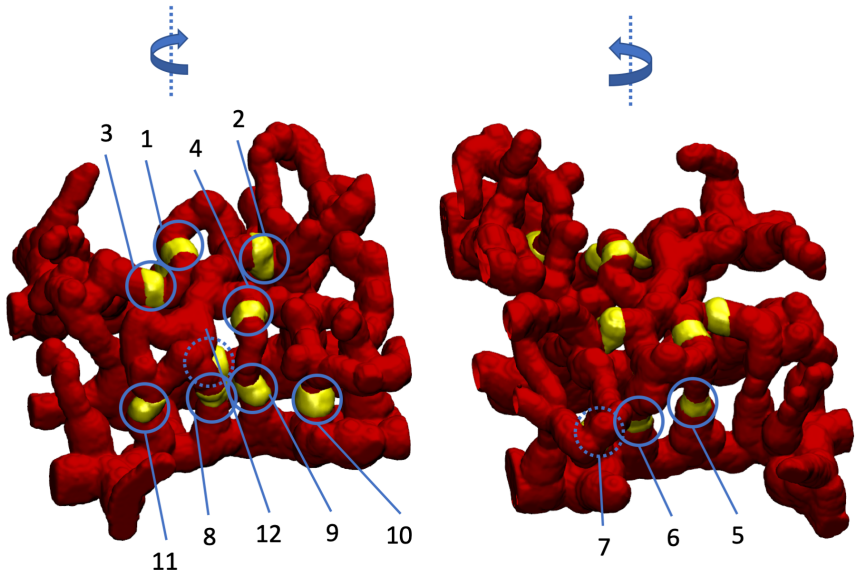


Figure 5.14: I present and analyze our results at 12 sites marked in the illustration. The sites are purposely spread out to sample the volume evenly.

5.3.2.3 Validation

I first simulate the heartbeat rhythm with the fine and the coarse solvers independently. The results are then compared at 12 sites. These sites are purposely spread out to sample the volume evenly, as marked in Figure 5.14. Since I assume the vessels cannot expand circumferentially, the volume-flux is directly related to the axial speed, which is presented in Figure 5.15.

The results from the standalone coarse solver (labeled as '*1D solver*') captures the flow under the Newtonian assumption. In addition, it ignores the effects of bifurcating angles in the bifurcating network. The fine solver, however, captures the vascular flow without those two simplifications. The results from the standalone fine solver are the reference solution that I consider to be accurate. Because the Non-Newtonian fluid is a mixture of RBC and plasma, the speed can fluctuate mildly. I measure the axial speed three times, one for each cardiac cycle, and plot the extreme

values in Figure 5.15.

From the first glance, I can see that the 1D (coarse) solution is strongly uniform in motion, as I expect from a non-stochastic coupled ODE solver. That is, the peaks are approximately in sync with little phase shift and with the inflow boundaries (Figure 5.13), whereas in reference solution the peak are out of sync at those sites. As the wave propagates across the domain from the inflow boundaries, the speed and magnitude depend on factors such as local fluid composition and branching angles. The fine solver models flow and fluid-solid interaction from the first principles, and accounts for those factors intrinsically. This would justify the differences at *Site 5, 6, 8* and *9*.

I then simulate the heartbeat with `SPASD`, and compare average speed against reference values at those sites. Here, I employ spatial decomposition (SD) in conjunction with `SPASD` for further acceleration. As explained in Section 5.3.2.2, `SPASD` solution must converge in less than four iterations to be beneficial. I therefore run three iterations and plotted the result against the reference solution in Figure 5.16. For each sub-domain in *Iteration 1*, the speed is initialized to match that of coarse (1D) solver. The initial "spikes" in each sub-domain correspond to the gap between the reference and coarse solution in Figure 5.15. The network flow then normalizes to the equilibrium as it recovers from the inaccurate initial value. The speed of recovery depends on factors such as the amount of deviation from the reference and magnitude of the reference. I want to remind the reader that the recovery is not a linear response. As a matter of fact, in complex networks with loops, a small local perturbation can have cascading reactions across the network and a lengthy equilibrium timeframe.

Iteration 2 is supposed to be initialized with values calculated via `SPASD`. Be-

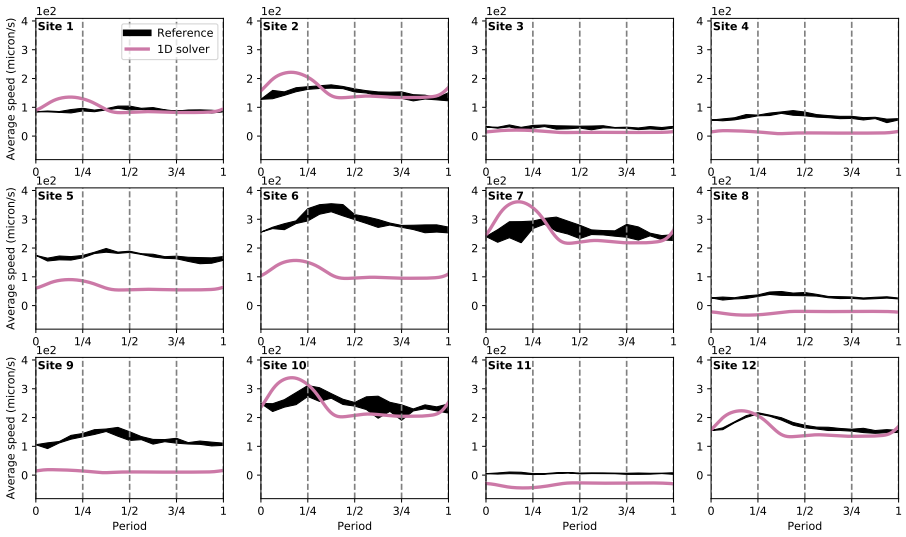


Figure 5.15: I simulated the heartbeat with CS (labeled as '*1D solver*') and FS (labeled as '*Reference*') separately. The resulting average speed at 12 sites marked in Figure 5.14 are shown here. The gap between *1D solver* and *Reference* is expected because of the difference in fidelity. FS models flow and fluid-solid interaction from the first principles, and accounts for local fluid composition and branching angles intrinsically.

cause the new values are within 10% threshold from that of last iteration, no re-initialization were carried out at those sites. Instead, FSs simply continue to march in time for the corresponding sub-domains. *Iteration 2* skips the first sub-domain since the solution is obtained via fine solver alone. It continues from where *Iteration 1* stops in *sub-domain 1*. The same goes for the other sub-domains. After only two iterations, the result is very close to the reference solution. I plot the reference solution and the result after two iterations in Figure 5.17. For demonstration purposes, I run a third iteration and plot it on the same figure. The difference after two and three iterations is less pronounced.

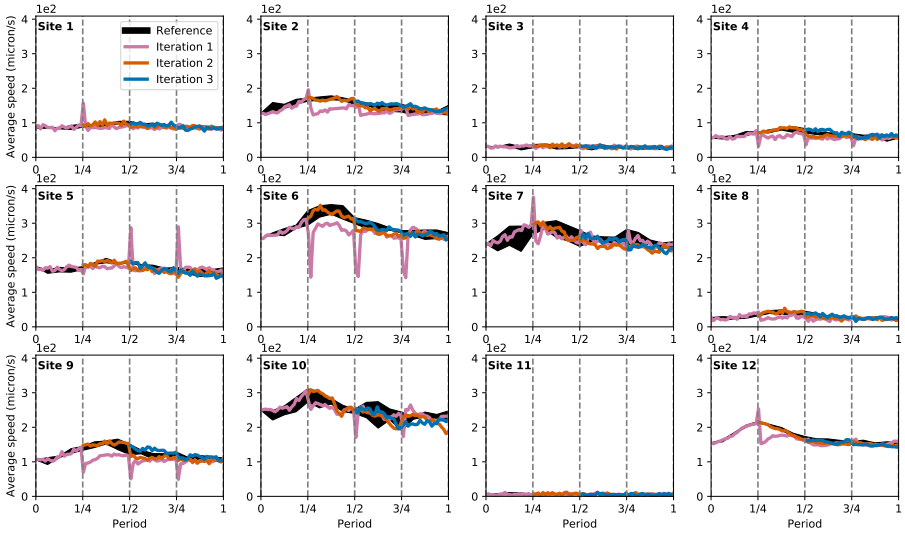


Figure 5.16: I simulate the heartbeat with `SPASD` for three iterations. The resulting average speed at 12 sites marked in Figure 5.14 are shown here. The reference solution from standalone FS simulation is also plotted for comparison. The initial "spikes" in each sub-domain correspond to the gap between the reference and coarse solution in Figure 5.15. The network flow then normalizes to the equilibrium as it recovers from the inaccurate initial value.

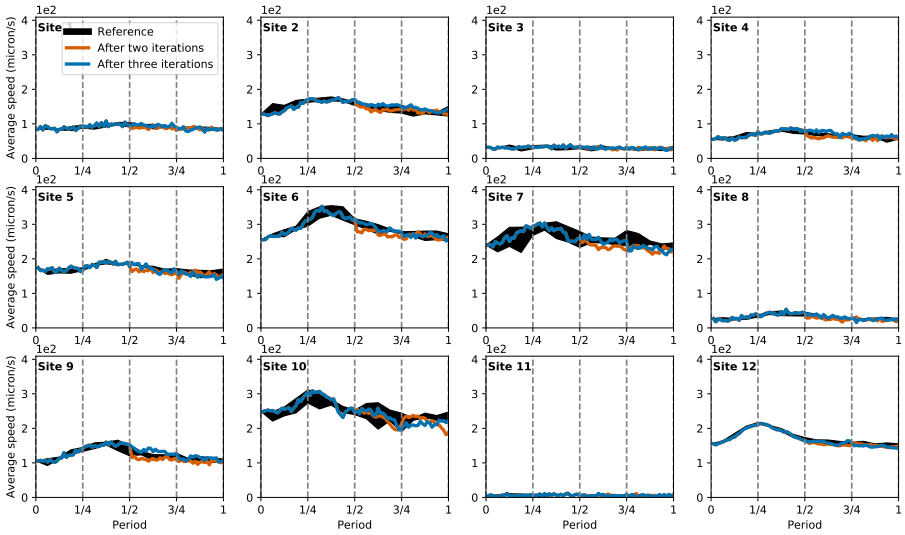


Figure 5.17: I consolidated the results in Figure 5.16 and re-plotted them here. After only two iterations, the result is very close to the reference solution. I run a third iteration for demonstration purposes.

5.4 Summary & Discussion

In this paper, I accelerated massive Lagrangian simulations of blood flow in zebrafish by employing the parallel-in-time algorithm `SPASD`. The framework was first demonstrated on a simple Y-bifurcation geometry. Physical quantities, i.e. velocity, shear rate, shear stress and volume-flux, were computed and compared against reference solutions. Specifically, the l^2 relative error on volume-flux is less than 1% for simple fluid and less than 3% for complex fluid on average. I then demonstrated our framework on a digital reconstruction of zebrafish hindbrain, which is a complex vascular network with 95 branches and 57 bifurcations. Comparatively, `SPASD` is better suited for simulating simple fluids than for complex fluids. As demonstrated in Y-bifurcation in Section 5.3.1, the volume-flux 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, the red blood cells are modeled explicitly in the complex fluid as a 2D triangular mesh. 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 volume-flux from `SPASD` and the reference in Figure 5.9. 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.

When simulating a large system with limited sources, SD offers better efficiency than `SPASD`. This does not contradict the limitations of SD listed in Section 5.1. Instead, this is drawn from our past benchmark on large-scale particle simulations, where I have demonstrated that SD scales very well with limited computational resources due to the absence of the coarse solver 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.

Lastly, I 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 an RBC in a future timestamp given its current location. To expand the statement, SPASD cannot predict the distribution of RBCs in the vescular network across time sub-domains. Instead, SPASD is more suitable for simulating well-mixed fluids. In the case of complex fluids such as blood, I are assuming the mixture of plasma and RBCs are homogeneous in the network scope. However, the homogeneous assumption is only used when computing volume-flux at the beginning and the end of each sub-domain, and it does not impact the integration of the fine solver inside a sub-domain.

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 depends on the trajectories of particles. These process usually involves accumulate lasting longer than one temporal sub-domains. In this case, decreasing the number of sub-domains and redirecting the compute resources to SD would be more beneficial.

5.5 Appendix

5.5.1 DPD Model Parameters

For the simple fluid example in Section 5.3.1.1, the DPD pair interaction parameters in Equation (5.3,5.4,5.5) are $\alpha = 30$, $\gamma = 4.5$, $\sigma = 3$, $s = 2$, $r_c = 1$.

		α	γ	σ	s	r_c
Wall	Any	60	45	2.916	2	1
Solvent	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

Table 5.6: DPD parameters of pair interaction in Y-bifurcation example.

		α	γ	σ	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

Table 5.7: DPD parameters of pair interaction in zebrafish hindbrain example.

For the complex fluid example in Section 5.3.1.2, the DPD pair interaction parameters in Equation (5.3,5.4,5.5) are tabulated in Table 5.6. For the simulation of flow in zebrafish hindbrain (Section 5.3.2), the parameters are tabulated in Table 5.7.

5.5.2 Mapping to Physical Units

For the DPD model, we set the length scale $[L^{\mathrm{DPD}}] = 1 \times 10^{-6}$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 $[T^{\mathrm{DPD}}]$ can be evaluated with relation:

$$[T^{\mathrm{DPD}}] = [L^{\mathrm{DPD}}] \frac{\nu^{\mathrm{P}}}{\nu^{\mathrm{M}}} \frac{\mu^{\mathrm{M}}}{\mu^{\mathrm{P}}} (\mathrm{s}), \tag{5.23}$$

where μ is the RBC membrane shear modulus, and ν 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 $\nu^{\mathrm{P}} = 1.2 \times 10^{-3}$ Pa s, and the RBC

membrane shear modulus $\mu_{\mathrm{s}}^{\mathrm{P}} = 4.5 \times 10^{-6}$ Nm^{-1}:

$$[T^{\mathrm{DPD}}] = 10^{-6} \frac{1.2 \times 10^{-3}}{21.8132} \frac{100}{4.5 \times 10^{-6}} = 1.2225 \times 10^{-3}\mathrm{s}. \tag{5.24}$$

Here, we set $\mu^{\mathrm{M}} = 100$ and measured $\nu^{\mathrm{M}} = 21.8132$ from running benchmark simulations.

Bibliography

[1] J. Alastruey, K. H. Parker, J. Peiró, and S. J. Sherwin. Lumped parameter outflow models for 1-D blood flow simulations: Effect on pulse waves and parameter estimation Design optimisation of tunnel ventilation systems View project Lumped Parameter Outflow Models for 1-D Blood Flow Simulations: Effect on Pulse W. *Commun. Comput. Phys.*, 4(2):317–336, 2008.

[2] J. A. Anderson, C. D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *J. Comput. Phys.*, 227(10):5342–5359, 2008. ISSN 00219991. doi: 10.1016/j.jcp.2008.01.047.

[3] M. Astorino, F. Chouly, and A. Quarteroni. Multiscale coupling of finite element and lattice Boltzmann methods for time dependent problems. *Politecnico di Milano*, 2012.

[4] J. Backer, C. Lowe, H. Hoefsloot, and P. Iedema. Poiseuille flow to measure the viscosity of particle model fluids. *J. Chem. Phys.*, 122(15):154503, 2005.

[5] J. A. Backer, C. P. Lowe, H. C. J. Hoefsloot, and P. D. Iedema. Poiseuille flow to measure the viscosity of particle model fluids. *J. Chem. Phys.*, 122(15):154503, 2005.

[6] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah. Parallel-in-time molecular-dynamics simulations. *Phys. Rev. E*, 66(5):057701, 2002.

[7] M. O. Bernabeu, M. L. Jones, J. H. Nielsen, T. Krüger, R. W. Nash, D. Groen, S. Schmieschek, J. Hetherington, H. Gerhardt, C. A. Franco, and P. V. Coveney. Computer simulations reveal complex distribution of haemodynamic forces in a mouse retina model of angiogenesis. *J. R. Soc. Interface*, 11(99):20140543–20140543, jul 2014. ISSN 17425662. doi: 10.1098/rsif.2014.0543.

[8] A. A. S. Bhagat, H. W. Hou, L. D. Li, C. T. Lim, and J. Han. Pinched flow coupled shear-modulated inertial microfluidics for high-throughput rare blood cell separation. *Lab Chip*, 11(11):1870–1878, 2011. ISSN 1473-0197. doi: 10.1039/c0lc00633e.

[9] A. Blouza, L. Boudin, S. M. Kaber, et al. Parallel in time algorithms with reduction methods for solving chemical kinetics. *Comm. App. Math. Com. Sc.*, 5(2):241–263, 2010.

[10] A. L. Blumers, Y.-H. Tang, Z. Li, X. Li, and G. E. Karniadakis. GPU-accelerated red blood cells simulations with transport dissipative particle dynamics. *Comput. Phys. Commun.*, 217:171–179, 2017.

[11] A. L. Blumers, Z. Li, and G. E. Karniadakis. Supervised parallel-in-time algorithm for long-time Lagrangian simulations of stochastic dynamics: Application to hydrodynamics. *J. Comput. Phys.*, may 2019. ISSN 0021-9991. doi: 10.1016/J.JCP.2019.05.016.

[12] E. Bouchbinder, T. Goldman, and J. Fineberg. The dynamics of rapid fracture: instabilities, nonlinearities and length scales. *Rep. Prog. Phys.*, 77(4):30, 2014.

[13] A. Brandt. Multiscale scientific computation: Review 2001. In *Multiscale and multiresolution methods*, pages 3–95. Springer, 2002.

[14] B. R. Brooks, C. L. Brooks III, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, and C. Bartels. CHARMM: The Biomolecular Simulation Program. *J. Comput. Chem.*, 31(16):2967–2970, 2010. ISSN 1096-987X. doi: 10.1002/jcc.

[15] E. J. Bylaska, J. Q. Weare, and J. H. Weare. Extending molecular simulation time scales: Parallel in time integrations for high-level quantum chemistry and complex force representations. *J. Chem. Phys.*, 139(7):074114, 2013.

[16] P. Carmeliet and R. K. Jain. Principles and mechanisms of vessel normalization for cancer and other angiogenic diseases. *Nat. Rev. Drug Discov.*, 10(6):417–27, jun 2011. ISSN 1474-1784. doi: 10.1038/nrd3455.

[17] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. The Amber biomolecular simulation programs. *J. Comput. Chem.*, 26(16):1668–88, dec 2005. ISSN 0192-8651. doi: 10.1002/jcc.20290.

[18] M. N. Chávez, G. Aedo, F. A. Fierro, M. L. Allende, and J. T. Egaña. Zebrafish as an Emerging Model Organism to Study Angiogenesis in Development and Regeneration. *Front. Physiol.*, 7:56, mar 2016. ISSN 1664-042X. doi: 10.3389/fphys.2016.00056.

[19] L. Coultas, K. Chawengsaksophak, and J. Rossant. Endothelial cells and VEGF in vascular development, dec 2005. ISSN 14764687.

[20] J. Crank. *The mathematics of diffusion*. Oxford University Press, London, 2nd revise edition, 1980.

[21] R. Elber. Perspective: Computer simulations of long time dynamics. *J. Chem. Phys.*, 144(6):060901, 2016.

[22] S. Engblom. Parallel in time simulation of multiscale stochastic chemical kinetics. *Multiscale Model. Sim.*, 8(1):46–68, 2009.

[23] R. Erban and S. J. Chapman. Reactive boundary conditions for stochastic simulations of reaction–diffusion processes. *Phys. Biol.*, 4(1):16–28, 2007.

[24] P. Español. Hydrodynamics from dissipative particle dynamics. *Phys. Rev. E*, 52(2):1732–1742, 1995.

[25] P. Espanol. Hydrodynamics from dissipative particle dynamics. *Phys. Rev. E*, 52(2):1734–1742, 1995. ISSN 1063651X. doi: 10.1103/PhysRevE.52.1734.

[26] P. Español and M. Revenga. Smoothed dissipative particle dynamics. *Phys. Rev. E.*, 67(2):26705, feb 2003. ISSN 1539-3755. doi: 10.1103/PhysRevE.67.026705.

[27] P. Español and P. Warren. Statistical mechanics of dissipative particle dynamics. *Europhys. Lett.*, 30(4):191–196, 1995. ISSN 0295-5075. doi: 10.1209/0295-5075/30/4/001.

[28] C. Farhat and M. Chandesris. Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications. *Int. J. Numer. Meth. Eng.*, 58(9):1397–1434, 2003.

[29] D. A. Fedosov, B. Caswell, and G. E. Karniadakis. A multiscale red blood cell model with accurate mechanics, rheology, and dynamics. *Biophys. J.*, 98(10):2215–2225, 2010. ISSN 00063495. doi: 10.1016/j.bpj.2010.02.002.

[30] K. A. Feenstra, B. Hess, and H. J. Berendsen. Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. *J. Comput. Chem.*, 20:786–798, 1999.

[31] K. A. Feenstra, B. Hess, and H. J. Berendsen. Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. *J. Comput. Chem.*, 20(8):786–798, jun 1999. ISSN 01928651. doi: 10.1002/(SICI)1096-987X(199906)20:8<786::AID-JCC5>3.0.CO;2-B.

[32] N. Filipovic, M. Kojic, and A. Tsuda. Modelling thrombosis using dissipative particle dynamics method. *Philos. Trans. Royal Soc. A*, 366(1879):3265–3279, 2008.

[33] N. Filipovic, M. Kojic, and A. Tsuda. Modelling thrombosis using dissipative particle dynamics method. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, 366(1879):3265–3279, sep 2008. ISSN 1364503X. doi: 10.1098/rsta.2008.0097.

[34] P. F. Fischer, F. Hecht, and Y. Maday. A parareal in time semi-implicit approximation of the Navier-Stokes equations. In *Domain decomposition methods in science and engineering*, pages 433–440. Springer, 2005.

[35] L. Formaggia, J.-f. Gerbeau, F. Nobile, A. Quarteroni, and S. J. Numer Anal. NUMERICAL TREATMENT OF DEFECTIVE BOUNDARY CONDITIONS FOR THE NAVIER-STOKES EQUATIONS *. *Soc. Ind. Appl. Math.*, 40(1):376–401, 2002.

[36] R. P. Franke, M. Gräfe, H. Schnittler, D. Seiffge, C. Mittermayer, and D. Drenckhahn. Induction of human vascular endothelial stress fibres by fluid shear stress. *Nature*, 307(5952):648–649, feb 1984. ISSN 00280836. doi: 10.1038/307648a0.

[37] G. Frantziskonis, K. Muralidharan, P. Deymier, S. Simunovic, P. Nukala, and S. Pannala. Time-parallel multiscale/multiphysics framework. *J. Comput. Phys.*, 228(21):8085–8092, 2009.

[38] P. L. Freddolino, C. B. Harrison, Y. Liu, and K. Schulten. Challenges in protein-folding simulations. *Nat. Phys.*, 6(10):751, 2010.

[39] I. Garrido, M. S. Espedal, and G. E. Fladmark. A convergent algorithm for time parallelization applied to reservoir simulation. In *Domain Decomposition Methods in Science and Engineering*, pages 469–476. Springer, 2005.

[40] A. W. Goetz, M. J. Williamson, D. Xu, D. Poole, S. L. Grand, and R. C. Walker. Routine Microsecond Molecular Dynamics Simulations with AMBER on GPUs. 1. Generalized Born. *J. Chem. Theory Comput.*, 8:1542—-1555., 2012.

[41] H. Grabert. *Projection Operator Techniques in Nonequillibrium Statistical Mechanics*. Springer Berlin Heidelberg, volume 95 edition, 1982.

[42] R. D. Groot and P. B. Warren. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. *J. Chem. Phys.*, 107(11): 4423, 1997. ISSN 00219606. doi: 10.1063/1.474784.

[43] J.-P. Hansen and I. McDonald. *Theory of Simple Liquids, Fourth Edition: with Applications to Soft Matter*. Academic Press, 2013.

[44] Y. Hasegawa and N. Kasagi. Low-pass filtering effects of viscous sublayer on high Schmidt number mass transfer close to a solid wall. *Int. J. Heat Fluid Flow*, 30(3):525–533, 2009.

[45] L. He. The reduced basis technique as a coarse solver for parareal in time simulations. *J. Comput. Math*, pages 676–692, 2010.

[46] C. Hijón, P. Español, E. Vanden-Eijnden, and R. Delgado-Buscalioni. Mori–Zwanzig formalism as a practical computational tool. *Faraday Discuss.*, 144(0):301–322, oct 2010. ISSN 1359-6640. doi: 10.1039/B902479B.

[47] B. Hogan, Z. Shen, H. Zhang, C. Misbah, and A. I. Barakat. Shear stress in the microvasculature: influence of red blood cell morphology and endothelial wall undulation. *Biomech. Model. Mechanobiol.*, pages 1–15, mar 2019. ISSN 16177940. doi: 10.1007/s10237-019-01130-8.

[48] S. H. Holm, J. P. Beech, M. P. Barrett, and J. O. Tegenfeldt. Separation of parasites from human blood using deterministic lateral displacement. *Lab Chip*, 11(7):1326–1332, 2011. ISSN 1473-0197. doi: 10.1039/c0lc00560f.

[49] P. J. Hoogerbrugge and J. M. V. A. Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *Eur. Lett*, 19(1): 155–160, 1992. ISSN 0295-5075. doi: 10.1209/0295-5075/19/3/001.

[50] G. Horton. The time-parallel multigrid method. *Commun. appl. numer. m.*, 8 (9):585–595, 1992.

[51] C. Huang, F. Sheikh, M. Hollander, C. Cai, D. Becker, P.-H. Chu, S. Evans, and J. Chen. Embryonic atrial function is essential for mouse embryogenesis, cardiac morphogenesis and angiogenesis. *Development*, 130(24):6111–6119, dec 2003. ISSN 0950-1991. doi: 10.1242/DEV.00831.

[52] M. Huang, Z. Li, and H. Guo. The effect of Janus nanospheres on the phase separation of immiscible polymer blends via dissipative particle dynamics simulations. *Soft Matter*, 8(25):6834, jun 2012. ISSN 1744-683X. doi: 10.1039/c2sm25086a.

[53] R. D. Jäggi, R. Sandoz, and C. S. Effenhauser. Microfluidic depletion of red blood cells from whole blood in high-aspect-ratio microchannels. *Microfluid. Nanofluidics*, 3(1):47–53, 2007. ISSN 16134982. doi: 10.1007/s10404-006-0104-9.

[54] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. NAMD2: Greater Scalability for Parallel Molecular Dynamics. *J. Comput. Phys.*, 151:283–312, 1999. ISSN 00219991. doi: 10.1006/jcph.1999.6201.

[55] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidid, O. Runborg, C. Theodoropoulos, et al. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Commun. Math. Sci.*, 1(4):715–762, 2003.

[56] M. Killer, A. S. Arthur, J. D. Barr, B. Richling, and G. M. Cruise. Histomorphology of thrombus organization, neointima formation, and foreign body response in retrieved human aneurysms treated with hydrocoil devices. *J. Biomed. Mater. Res. B*, 94B(2):486–492, 2010.

[57] M. Killer, A. S. Arthur, J. D. Barr, B. Richling, and G. M. Cruise. Histomorphology of thrombus organization, neointima formation, and foreign body response in retrieved human aneurysms treated with hydrocoil devices. *J. Biomed. Mater. Res. - Part B Appl. Biomater.*, 94(2):486–492, jun 2010. ISSN 15524973. doi: 10.1002/jbm.b.31660.

[58] T. Kinjo and S.-a. Hyodo. Equation of motion for coarse-grained simulation based on microscopic description. *Phys. Rev. E*, 75(5):051109, may 2007. ISSN 1539-3755. doi: 10.1103/PhysRevE.75.051109.

[59] J. M. V. A. Koelman and P. J. Hoogerbrugge. Dynamic Simulations of Hard-Sphere Suspensions Under Steady Shear. *Europhys. Lett.*, 21(3):363–368, jan 1993. ISSN 0295-5075. doi: 10.1209/0295-5075/21/3/018.

[60] R. Kopp, T. Schwerte, and B. Pelster. Cardiac performance in the zebrafish breakdance mutant. *J. Exp. Biol.*, 208(Pt 11):2123–34, jun 2005. ISSN 0022-0949. doi: 10.1242/jeb.01620.

[61] J. Kordilla, W. Pan, and A. Tartakovsky. Smoothed particle hydrodynamics model for Landau-Lifshitz-Navier-Stokes and advection-diffusion equations. *J. Chem. Phys.*, 141(22), 2014. ISSN 00219606. doi: 10.1063/1.4902238.

[62] J. Lee, T.-C. Chou, D. Kang, H. Kang, J. Chen, K. I. Baek, W. Wang, Y. Ding, D. D. Carlo, Y.-C. Tai, and T. K. Hsiai. A Rapid Capillary-Pressure Driven Micro-Channel to Demonstrate Newtonian Fluid Behavior of Zebrafish Blood at High Shear Rates. *Sci. Rep.*, 7(1):1980, dec 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-02253-7.

[63] F. Legoll, T. Lelievre, and G. Samaey. A micro-macro parareal algorithm: application to singularly perturbed ordinary differential equations. *SIAM J. Sci. Comput.*, 35(4):A1951–A1986, 2013.

[64] H. Lei, B. Caswell, and G. E. Karniadakis. Direct construction of mesoscopic models from microscopic simulations. *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, 81(2):1–10, 2010. ISSN 15393755. doi: 10.1103/PhysRevE.81.026704.

[65] H. Li, V. Ha, and G. Lykotrafitis. Modeling sickle hemoglobin fibers as one chain of coarse-grained particles. *J. Biomech.*, 45(11):1947–1951, jul 2012. ISSN 00219290. doi: 10.1016/j.jbiomech.2012.05.016.

[66] X. Li, P. M. Vlahovska, and G. E. Karniadakis. Continuum- and particle-based modeling of shapes and dynamics of red blood cells in health and disease. *Soft Matter*, 9(1):28–37, 2013. ISSN 1744-6848. doi: 10.1039/C2SM26891D.

[67] X. Li, Z. Peng, H. Lei, M. Dao, and G. E. Karniadakis. Probing red blood cell mechanics, rheology and dynamics with a two-component multi-scale model. *Phil. Trans. R. Soc. A*, 372(2021):20130389—-, 2014. ISSN 1364-503X. doi: 10.1098/rsta.2013.0389.

[68] X. Li, Y.-H. Tang, H. Liang, and G. E. Karniadakis. Large-scale dissipative particle dynamics simulations of self-assembled amphiphilic systems. *Chem. Commun. (Camb).*, 50(61):8306–8, 2014. ISSN 1364-548X. doi: 10.1039/c4cc03096f.

[69] Z. Li, X. Bian, B. Caswell, and G. E. Karniadakis. Construction of dissipative particle dynamics models for complex fluids via the Mori-Zwanzig formulation. *Soft Matter*, 10(43):8659–8672, 2014.

[70] Z. Li, X. Bian, B. Caswell, and G. E. Karniadakis. Construction of dissipative particle dynamics models for complex fluids via the Mori-Zwanzig formulation. *Soft Matter*, 10(43):8659–8672, 2014. ISSN 1744-6848. doi: 10.1039/c4sm01387e.

[71] Z. Li, Y.-H. Tang, H. Lei, B. Caswell, and G. E. Karniadakis. Energy-conserving dissipative particle dynamics with temperature-dependent properties. *J. Comput. Phys.*, 265:113–127, 2014.

[72] Z. Li, Y.-H. H. Tang, H. Lei, B. Caswell, and G. E. Karniadakis. Energy-conserving dissipative particle dynamics with temperature-dependent properties. *J. Comput. Phys.*, 265:113–127, 2014. ISSN 00219991. doi: 10.1016/j.jcp.2014.02.003.

[73] Z. Li, X. Bian, X. Li, and G. E. Karniadakis. Incorporation of memory effects in coarse-grained modeling via the Mori-Zwanzig formalism. *J. Chem. Phys.*, 143(24):243128, 2015.

[74] Z. Li, Y.-H. Tang, X. Li, and G. E. Karniadakis. Mesoscale modeling of phase transition dynamics of thermoresponsive polymers. *Chem. Commun.*, 51:4–6, 2015. ISSN 1359-7345. doi: 10.1039/C5CC01684C.

[75] Z. Li, A. Yazdani, A. Tartakovsky, and G. E. Karniadakis. Transport dissipative particle dynamics model for mesoscopic advection-diffusion-reaction problems. *J. Chem. Phys.*, 143(1):014101, 2015. ISSN 00219606. doi: 10.1063/1.4923254.

[76] Z. Li, X. Bian, X. Li, M. Deng, Y.-H. Tang, B. Caswell, and G. E. Karniadakis. Dissipative particle dynamics: foundation, evolution, implementation, and applications. In *Particles in Flows*, pages 255–326. Springer, 2017.

[77] Z. Li, X. Bian, Y.-H. Tang, and G. E. Karniadakis. A solution to arbitrary-shaped geometries in dissipative particle dynamics (Manuscript). *J. Comput. Phys.*, 2017.

[78] Z. Li, X. Bian, Y.-H. Tang, and G. E. Karniadakis. A dissipative particle dynamics method for arbitrarily complex geometries. *J. Comput. Phys.*, 355: 534–547, 2018.

[79] G. J. Lieschke and P. D. Currie. Animal models of human disease: Zebrafish swim into view, may 2007. ISSN 14710056.

[80] H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm. ESPResSo-an extensible simulation package for research on soft matter systems. *Comput. Phys. Commun.*, 174(9):704–727, 2006. ISSN 00104655. doi: 10.1016/j.cpc.2005.10.005.

[81] K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw. How Fast-Folding Proteins Fold. *Science*, 334(6055):517–520, 2011. ISSN 0036-8075.

[82] J. L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps «pararéel ». *Comptes Rendus l'Academie des Sci. - Ser. I Math.*, 332 (7):661–668, apr 2001. ISSN 07644442. doi: 10.1016/S0764-4442(00)01793-6.

[83] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps «pararéel». *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(7):661–668, 2001.

[84] C. A. Marsh and J. M. Yeomans. Dissipative particle dynamics: The equilibrium for finite time steps. *Europhys. Lett.*, 37(8):511–516, mar 1997. ISSN 0295-5075. doi: 10.1209/epl/i1997-00183-2.

[85] C. A. Marsh, G. Backx, and M. H. Ernst. Static and dynamic properties of dissipative particle dynamics. *Phys. Rev. E*, 56(2):1676–1691, 1997.

[86] Z. G. Mills, W. Mao, and A. Alexeev. Mesoscale modeling: solving complex flows in biology and biotechnology. *Trends Biotechnol.*, 31(7):426–434, jul 2013. ISSN 0167-7799. doi: 10.1016/J.TIBTECH.2013.05.001.

[87] S. Mitran. Time parallel kinetic-molecular interaction algorithm for CPU/GPU computers. *Procedia. Comput. Sci.*, 1(1):745–752, 2010.

[88] J. J. Monaghan. Smoothed Particle Hydrodynamics. *World Sci.*, 2003.

[89] H. Mori. Transport, Collective Motion, and Brownian Motion. *Prog. Theor. Phys.*, 33(3):423–455, mar 1965. ISSN 0033-068X. doi: 10.1143/PTP.33.423.

[90] J. M. Murtha, W. Qi, and E. T. Keller. Hematologic and serum biochemical values for zebrafish (Danio rerio). *Comp. Med.*, 53(1):37–41, 2003. ISSN 1532-0820.

[91] X. Nie, M. O. Robbins, and S. Chen. Resolving Singular Forces in Cavity Flow: Multiscale Modeling from Atomic to Millimeter Scales. *Phys. Rev. Lett.*, 96: 134501, 2006.

[92] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Commun. ACM.*, 7(12):731–733, 1964.

[93] J. Ortiz de Zárate and J. Sengers. *Hydrodynamic fluctuations in fluids and fluid mixtures*. Elsevier Science, Amsterdam, 2006.

[94] O. Oulaid and J. Zhang. Temporal and Spatial Variations of Wall Shear Stress in the Entrance Region of Microvessels. *J. Biomech. Eng.*, 137(6):061008, jun 2015. ISSN 0148-0731. doi: 10.1115/1.4030055.

[95] A. Pérez, F. J. Luque, and M. Orozco. Frontiers in molecular dynamics simulations of DNA. *Acc. Chem. Res.*, 45(2):196–205, 2012.

[96] N. Phan-Thien, N. Mai-Duy, and B. C. Khoo. A spring model for suspended particles in dissipative particle dynamics. *J. Rheol. (N. Y. N. Y).*, 58(4):839–867, jul 2014. ISSN 0148-6055. doi: 10.1122/1.4874679.

[97] M. Z. Pindera, H. Ding, M. M. Athavale, and Z. Chen. Accuracy of 1D microvascular flow models in the limit of low Reynolds numbers. *Microvasc. Res.*, 77(3):273–280, may 2009. ISSN 0026-2862. doi: 10.1016/J.MVR.2008.11.006.

[98] I. Pivkin, P. Richardson, and G. E. Karniadakis. Effect of red blood cells on platelet aggregation. *IEEE Eng. Med. Biol. Mag.*, 28(2):32–37, 2009.

[99] I. V. Pivkin and G. E. Karniadakis. Accurate coarse-grained modeling of red blood cells. *Phys. Rev. Lett.*, 101(11):1–4, 2008. ISSN 00319007. doi: 10.1103/PhysRevLett.101.118105.

[100] I. V. Pivkin, P. D. Richardson, and G. E. Karniadakis. Effect of red blood cells on platelet aggregation. *IEEE Eng. Med. Biol. Mag.*, 28(2009):32–7, 2009. ISSN 1937-4186. doi: 10.1109/MEMB.2009.931788.

[101] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics, mar 1995. ISSN 00219991.

[102] M. Prigozhin and M. Gruebele. Microsecond folding experiments and simulations: a match is made. *Phys. Chem. Chem. Phys.*, 15(10):3372–3388, 2013.

[103] 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, and E. Lindahl. GROMACS 4.5: A high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7):845–854, 2013. ISSN 13674803. doi: 10.1093/bioinformatics/btt055.

[104] A. Quarteroni and A. Veneziani. ANALYSIS OF A GEOMETRICAL MULTISCALE MODEL BASED ON THE COUPLING OF ODEs AND PDEs FOR BLOOD FLOW SIMULATIONS *. *Proc. Eur. Congr. Comput. Methods Appl. Sci. Eng.*, 1(2):359–364, 2002.

[105] N. Resnick and M. A. Gimbrone. Hemodynamic forces are complex regulators of endothelial gene expression. *FASEB J.*, 9(10):874–82, jul 1995. ISSN 0892-6638. doi: 10.1096/fasebj.9.10.7615157.

[106] W. Risau. Mechanisms of angiogenesis. *Nature*, 386(6626):671–674, apr 1997. ISSN 0028-0836. doi: 10.1038/386671a0.

[107] Robert Zwanzig. *Nonequilibrium statistical mechanics*. Oxford Uni- versity Press, volume 54 edition, 2001.

[108] 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, Y.-H. Tang, K. Lykov, D. Alexeev, M. Bernaschi, P. Hadjidoukas, M. Bisson, W. Joubert, and C. Conti. The in-silico lab-on-a-chip: petascale and high-throughput simulations of microfluidics at cell resolution. *Proc. Int. Conf. High Perform. Comput. Networking, Storage Anal. - SC '15*, pages 1–12, 2015. ISSN 21674337. doi: 10.1145/2807591.2807677.

[109] D. Samaddar, D. E. Newman, and R. Sánchez. Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm. *J. Comput. Phys.*, 229(18):6558–6573, 2010.

[110] A. J. Schriefl, M. J. Collins, D. M. Pierce, G. A. Holzapfel, L. E. Niklason, and J. D. Humphrey. Remodeling of intramural thrombus and collagen in an Ang-II infusion ApoE-/- model of dissecting aortic aneurysms. *Thromb. Res.*, 130(3):e139–e146, 2012.

[111] A. J. Schriefl, M. J. Collins, D. M. Pierce, G. A. Holzapfel, L. E. Niklason, and J. D. Humphrey. Remodeling of intramural thrombus and collagen in an Ang-II infusion ApoE-/-model of dissecting aortic aneurysms. *Thromb. Res.*, 130 (3):e139–e146, sep 2012. ISSN 00493848. doi: 10.1016/j.thromres.2012.04.009.

[112] T. W. Secomb, J. P. Alberding, R. Hsu, M. W. Dewhirst, and A. R. Pries. Angiogenesis: An Adaptive Dynamic Biological Patterning Problem. *PLoS Comput. Biol.*, 9(3):e1002983, mar 2013. ISSN 1553734X. doi: 10.1371/journal.pcbi.1002983.

[113] S. Sherwin, V. Franke, J. Peiró, and K. Parker. One-dimensional modelling of a vascular network in space-time variables. *J. Eng. Math.*, 47(3/4):217–250, dec 2003. ISSN 0022-0833. doi: 10.1023/B:ENGI.0000007979.32871.e2.

[114] S. J. Sherwin, L. Formaggia, J. Peiró, and V. Franke. 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. Methods Fluids*, 43(6-7):673–700, oct 2003. ISSN 02712091. doi: 10.1002/fld.543.

[115] D. N. Sibley, A. Nold, N. Savva, and S. Kalliadasis. On the moving contact line singularity: Asymptotics of a diffuse-interface model. *Eur. Phys. J. E*, 36 (3):26, 2013.

[116] L. D. G. Sigalotti, J. Klapp, E. Sira, Y. Meleán, and A. Hasmy. SPH simulations of time-dependent Poiseuille flow at low Reynolds numbers. *J. Comput. Phys.*, 191(2):622–638, 2003.

[117] W. Smith and T. R. Forester. DL_POLY_2.0: A general-purpose parallel molecular dynamics simulation package. *J. Mol. Graph.*, 14(3):136–141, 1996. ISSN 02637855. doi: 10.1016/S0263-7855(96)00043-4.

[118] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon. A massively space-time parallel N-body solver. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 92. IEEE Computer Society Press, 2012.

[119] S. Suresh, J. Spatz, J. P. Mills, A. Micoulet, M. Dao, C. T. Lim, M. Beil, and T. Seufferlein. Connections between single-cell biomechanics and human disease states: gastrointestinal cancer and malaria. *Acta Biomater.*, 23(S): S3—-S15, 2015. ISSN 18787568. doi: 10.1016/j.actbio.2015.07.015.

[120] Y.-H. Tang, S. Kudo, X. Bian, Z. Li, and G. E. Karniadakis. Multiscale Universal Interface: A concurrent framework for coupling heterogeneous solvers. *J. Comput. Phys.*, 297:13–31, 2015. ISSN 00219991. doi: 10.1016/j.jcp.2015.05.004.

[121] Y.-H. Tang, S. Kudo, X. Bian, Z. Li, and G. E. Karniadakis. Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers. *J. Comput. Phys.*, 297:13–31, 2015.

[122] Y.-H. H. Tang and G. E. Karniadakis. Accelerating dissipative particle dynamics simulations on GPUs: Algorithms, numerics and applications. *Comput. Phys. Commun.*, 185(11):2809–2822, 2014. ISSN 00104655. doi: 10.1016/j.cpc.2014.06.015.

[123] Y.-H. H. Tang, Z. Li, X. Li, M. Deng, and G. E. Karniadakis. Non-equilibrium dynamics of vesicles and micelles by self-assembly of block copolymers with double thermoresponsivity. *Macromolecules*, 49(7):2895–2903, 2016. ISSN 15205835. doi: 10.1021/acs.macromol.6b00365.

[124] A. P. Thompson, S. J. Plimpton, and W. Mattson. The Statistical Mechanical Theory of Transport Processes. IV. The Equations of Hydrodynamics The Journal of Chemical Physics. *J. Chem. Phys.*, 131:926, 2009. doi: 10.1063/1.3245303.

[125] E. Weinan, B. Engquist, and Z. Huang. Heterogeneous multiscale method: a general methodology for multiscale modeling. *Phys. Rev. B*, 67(9):092101, 2003.

[126] N. Westerhof, J.-W. Lankhaar, and B. E. Westerhof. The arterial windkessel. *Medical & biological engineering & computing*, 47(2):131–141, 2009.

[127] W. Xiong and J. Zhang. Shear stress variation induced by red blood cell motion in microvessel. *Ann. Biomed. Eng.*, 38(8):2649–2659, aug 2010. ISSN 00906964. doi: 10.1007/s10439-010-0017-3.

[128] Q. Xu, J. S. Hesthaven, and F. Chen. A parareal method for time-fractional differential equations. *J. Comput. Phys.*, 293:173–183, 2015.

[129] Z. Xu, P. Meakin, A. Tartakovsky, and T. D. Scheibe. Dissipative-particle-dynamics model of biofilm growth. *Phys. Rev. E*, 83(6):1–9, 2011. ISSN 15393755. doi: 10.1103/PhysRevE.83.066702.

[130] A. Yazdani, H. Li, J. D. Humphrey, and G. E. Karniadakis. A general shear-dependent model for thrombus formation. *PLOS Comput. Biol.*, 13(1): e1005291, 2017.

[131] S. Yip and M. P. Short. Multiscale materials modelling at the mesoscale. *Nature materials*, 12(9):774, 2013.

[132] A. Zakrzewicz, T. W. Secomb, and A. R. Pries. Angioadaptation: Keeping the Vascular System in Shape. *Physiology*, 17(5):197–201, oct 2015. ISSN 1548-9213. doi: 10.1152/nips.01395.2001.

[133] L. Zang, Y. Shimada, Y. Nishimura, T. Tanaka, and N. Nishimura. A Novel, Reliable Method for Repeated Blood Collection from Aquarium Fish. *Zebrafish*, 10(3):425–432, sep 2013. ISSN 1545-8547. doi: 10.1089/zeb.2012.0862.

[134] R. Zwanzig. Memory Effects in Irreversible Thermodynamics. *Phys. Rev.*, 124 (4):983–992, nov 1961. ISSN 0031-899X. doi: 10.1103/PhysRev.124.983.

[135] R. Zwanzig. *Nonequilibrium statistical mechanics*. Oxford University Press, New York, 2001.