LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-PROC-644182

A Spatio-Temporal Coupling Method to Reduce the Time-to-Solution of Cardiovascular Simulations

A. E. Randles, E. Kaxiras

September 24, 2013

IEEE International Parallel & Distributed Processing Symposium
Phoenix, AZ, United States
May 19, 2014 through May 23, 2014

Disclaimer

A Spatio-Temporal Coupling Method to Reduce the Time-to-Solution of Cardiovascular Simulations

Amanda Randles
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, USA
randles2@llnl.gov

Efthimios Kaxiras
Department of Physics and
School of Engineering and Applied Sciences
Harvard Universty
Cambridge, MA, USA
kaxiras@seas.harvard.edu

***Abstract*—We present a new parallel-in-time method designed to reduce the overall time-to- solution of a patient-specific cardiovascular flow simulation. Using a modified parareal algorithm, our approach extends strong scalability beyond spatial parallelism with fully controllable accuracy and no decrease in stability. We discuss the coupling of spatial and temporal domain decompositions used in our implementation, and showcase the use of the method on a study of blood flow through the aorta. We observe an additional 40% reduction in overall wall clock time with no significant loss of accuracy, in agreement with a predictive performance model.**

***Keywords*-computational fluid dynamics; parallel-in-time; lattice Boltzmann;**

I. Introduction

Computational fluid dynamic (CFD) simulations of biological flows are used for identification of regions of the circulatory system at risk for the development and progression of heart disease and have helped yield deep insights into the underlying mechanisms that experimental measurements alone could not have achieved (e.g. [1], [2], [3], [4], [5]). Because some circulatory phenomenon manifest over relatively long time scales, reducing the time-to-solution of these simulations is an important challenge. Reducing the run time of these simulations is a more challenging problem than enabling the modeling of larger fluid systems. There is a large amount of prior art regarding the exploitation of spatial parallelism that can be leveraged when increasing the domain size of the simulated fluid that is not available when extending a simulation in time (c.f. [6], [7], [8], [9], [10]). For many fluid simulations, the parallel efficiency saturates as soon as the size of the fluid domain drops below a certain threshold. After this point, adding additional cores no longer improves the overall time-to-solution. Decomposing the problem in the temporal domain as well as the spatial can assist in overcoming this inherent strong scaling limit and significantly reduce the wallclock time if sufficient computational resources are available.

This paper presents a method for reducing the overall time-to-solution of a patient-specific cardiovascular hemodynamics simulation that achieves high-accuracy results using a coupling decomposition process. Unlike previous work, our approach exploits both temporal and spatial domain decomposition while maintaining a stable explicit fluid solver. We present initial performance and error analysis of applying the method to the computational hemodynamics application, HARVEY [11]. We will focus on an adaptation of the parareal algorithm first introduced by Lions *et al.*, which combines independent coarse and fine resolutions in time to reduce the wallclock time of real time problems [12]. The fine representation is more computationally expensive and is run in parallel on multiple time intervals to refine the result of that individual interval. The result for the coarse resolution is calculated serially and used to initialize the fine representation, which is in turn calculated in parallel. The coupling of the two iterators provides a predictor-corrector scheme that iteratively refines the initial values of the fine solver and completes the refinement (correction step) in parallel. The algorithm consists of a series of these iterations that reach completion when the results converge within a set tolerance.

One of the main contributions in this paper is the development of a multilevel spatio-temporal coupling (MSTC) that enforces a hierarchical decomposition of the default communicator and allows seamless communication between the spatial and temporal decompositions. Introducing such a scheme into a lattice Boltzmann code such as HARVEY requires computational and algorithmic developments to address challenges related to the accuracy, scalability, and stability of coupling these two fluid representations. We demonstrate the ability of our method to not only efficiently produces accurate results, but to recover time-dependent phenomena like the pulsatile flow imposed by the beating of the heart. Using 65,536 cores of the IBM Blue Gene/Q supercomputer, we show a strong correlation between the predicted and observed parallel speedup and achieve a 40% reduction in runtime as compared to the optimized spatial scaling result.

This work performed under the auspices of the DOE by LLNL under Contract DE-AC52-07NA27344.

II. Related Work

Parallel-in-time methods have been investigated as ways to go beyond the strong-scaling limit of many applications. If the hardware is available, the combination of a coarse and fine solver can converge and enable a shorter time-to-solution for models of interest. The parareal algorithm was first proposed in 2001 [12] and solidified in the predictor-corrector format shortly after (e.g. [13], [14]). It converges with the same accuracy as would be achieved with the expensive or fine iterator. Recently, alternate parallel-in-time methods have been propose such as the parallel implicit time-integration algorithms (PITA) in which parallelizes the time-loop of a time-dependent PDE solver without impacting the serial compo (c.f. [15],[16]). Other studies use intertwining the iterations of parareal with the spectral deferred corrections, the Parallel Full Approximation Scheme in Space and Time (PFASST) ([17], [18]). Speck *et al.* demonstrated a speedup of $8\times$ in addition to the spatial speedup on up to 294,912 cores [19].

In this paper, we focus on adapting the parareal algorithm to enable space-time parallel simulation of computational fluid dynamics (CFD). Previous work has shown the potential for such algorithms in CFD applications like in recovering time dependent behavior such as the development of turbulent flow [20], however, most of the literature emphasizes the algorithmic aspects of the time-parallel schemes and there are only a few examples of studies into the efficiency of spatio-temporal coupling. There has been some research into the use of the parareal algorithm combined with spatial decomposition for a Navier-Stokes solver on up to 2,048 cores modeling flow passed a cylinder [21], but there are no studies that we are aware of investigating the coupling of spatial and temporal parallelism at larger scales for CFD for complex flow in a real 3 dimensional problem.

Here to model patient specific cardiovascular hemodynamics, we employ the lattice Boltzmann method (LBM), which we will discuss in the following section.

III. The Lattice Boltzmann Method

Heart disease is still one of the leading causes of death in the western world. In 50% of these cases, sudden cardiac death is the first manifestation of the disease [22]. Over the last few decades, physicians have linked key properties to the likely development and progression of heart disease but finding methods to identify and track these quantities for individual patients remains an outstanding question. There is a growing literature base of studies of modeling hemodynamic flows in patient-specific arterial geometries (c.f. [3], [6]); however, the need to model both a large fluid system and a long time domain poses significant challenges.

Our approach is based an algorithm that can efficiently model flow through complex geometries such as those found in the coronary arteries or the aorta. In order to capture the flow patterns accurately and efficiently, it is necessary to use a method that handles complex boundaries well. To this end, we use an alternative to the traditional Navier-Stokes equations, the lattice Boltzmann method (LBM) ([23], [24]). The LBM is a low-Mach, weakly compressible solver that recovers hydrodynamic behavior in the limit of small Knudsen numbers. A key advantage is the use of simplified kinetic models that macroscopic quantities like shear stress and pressure can be calculated without the need of body-fitted grid or an expensive Poisson solver. Such explicit finite difference methods are natural to parallelize and easy to implement at the tradeoff of small time steps and use of a high resolution discrete grid. The high level of scalability possible on massively parallel systems with the LBM (e.g. [7], [25], [26], [27]) makes it a strong option for large patient-specific blood flow simulations. Relying on such an explicit solver enables a the implementation of a spatio-temporal decomposition scheme as will be discussed in later sections.

The LB formalism comes from kinetic theory and is a minimal form of the Boltzmann equation based on the collective dynamics of fictitious particles that represent a local ensemble of molecules moving between the points of a regular Cartesian lattice. The fundamental quantity is the particle distribution function, denoted $f_i(\vec{x},t)$, which represents the probability of finding particles traveling with velocity ξ at lattice node x and at time t. The velocity space is discretized and the fluid dynamics are resolved through the evolution of $f_i(\vec{x},t)$ with time as:

$$f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) = f_i(\vec{x},t) - \omega\Delta t[f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)] \tag{1}$$

where $f_i^{eq}(\vec{x},t)$ is the equilibrium distribution and ω is the dimensionless relaxation parameter (related to the frequency of particle collisions) [24]. In this work, we use the 19-speed cubic D3Q19 lattice connecting each lattice point to its first and second neighbors [28]. There are two key components to the algorithm: advection and collision. The advection step propagates the fluid particles along the discretized velocity paths defined by the lattice.

Collisions are calculated via a relaxation toward local equilibrium, as illustrated in the right hand side of Eq.(1). We use the Bhatnagar-Gross-Krook (BGK) collision operator with a single relaxation time scale [29]. The local equilibrium is the result of a second-order expansion in the fluid velocity of a local Maxwellian with speed $\vec{u}$ and is defined by:

$$f_i^{eq} = w_i\rho\left[1 + \frac{\vec{c}_i\cdot\vec{u}}{c_s^2} + \frac{1}{2}\left(\frac{(\vec{c}_i\cdot\vec{u})^2}{(c_s^2)^2} - \frac{u^2}{c_s^2}\right)\right] \tag{2}$$

where rho denotes the density, $\vec{u}$ the average fluid speed, c_s the speed of sound in the lattice, and w_i the weights attributed to each discretized velocity as determined by the lattice structure. A no-slip boundary condition is enforced at the walls through a full bounce back scheme.

While all of the key data such as density, velocity, and pressure can be calculated based only on data from nearest neighbors, one drawback is that the LBM requires many small time steps as limited by CFL-type conditions. These time steps, however, are extremely efficient and make the method amenable for parallel implementations [11].

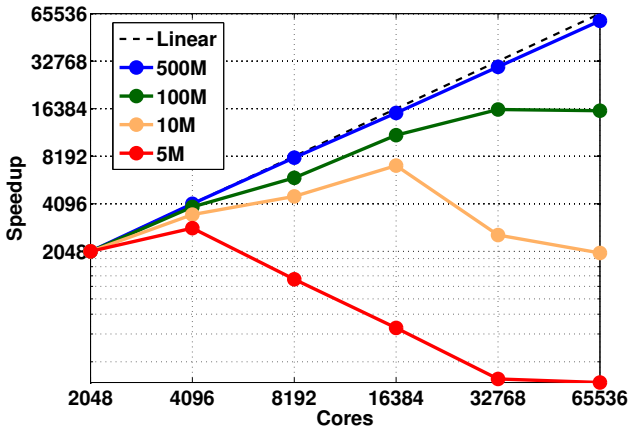

Figure 1. Speedup of LBM simulations using HARVEY for a range of fluid system sizes on up to 65,536 cores of the IBM Blue Gene/Q supercomputer.

A. Spatial Scaling Limit

While the LBM has been shown to scale with high efficiency up to 294,912 cores [7], there is a limit to the strong scaling potential when using only spatial decomposition. As more cores are used, fewer and fewer fluid nodes are allocated per code. As this ratio diminishes, the cost of the internode communication starts to overwhelm the runtime and reduce the scaling efficiency. In this work, we employed an even spatial decomposition in which the bounding box of the fluid was broken up into small cubes of equal sizes by evenly splitting each dimension. Fig. 1 shows the speedup achieved on up to 65,636 cores of the IBM Blue Gene/Q supercomputer at Livermore National Laboratory for fluid systems ranging in size from 5-100 million fluid nodes. As shown, for fixed-size fluid systems, beyond a certain point, adding more cores has no benefit and can actually slow down the simulation. As the number of fluid nodes per core drops below 5000, the time spent in communication starts to overwhelm the computation resulting in lower overall speedup. Fig. 2 highlights this drop-off showing a decrease in parallel efficiency for three different LBM codes. All of the scaling studies were completed on IBM Blue Gene/P supercomputers. The data for the second two applications were obtained from [7] and [9] respectively. While the second two codes included red blood cell modeling as well as the CFD component, all three demonstrate the same overall decline in efficiency corresponding to the number of fluid nodes per core. This is the intrinsic spatial scaling limit and will serve as the baseline for our parallel spatio-temporal simulations.

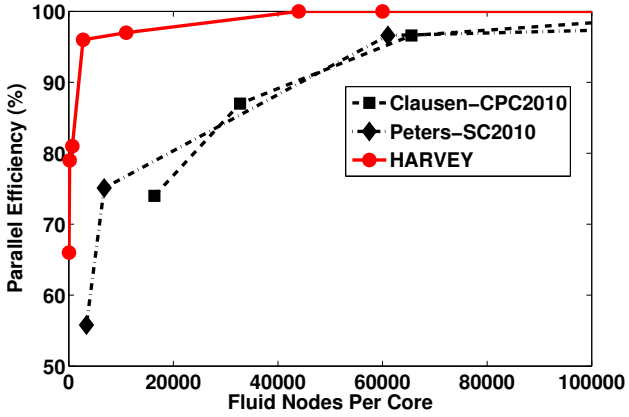

Figure 2. Parallel efficiency in terms of cores per fluid node for three different lattice Boltzmann codes. HARVEY is the application presented here. The other two codes include red blood models and the scaling studies were also completed on IBM Blue Gene/P supercomputers (c.f. [7], [9]).

IV. Parareal Algorithm

To achieve temporal decomposition, the entire time interval to be covered by the simulation is divided into N separate intervals of equal size, $[t_{n-1}, t_n], n = 1 \ldots N$, with n referring to the n^{th} time step. If there is no spatial parallelization, N is set to the number of cores in the system. An iteratively refined estimation of the result for each interval is calculated, denoted U_{n+1}^K are calculated where K is the parareal iteration number. For each parareal iteration, K, U_n^0 is initialized through the serial application of the coarse solver for the full time domain, $0 \ldots, t_N$. For each $K > 0$, the initial estimation, U_n^0, for the time interval handled be each processor is refined by simultaneously running the fine solver for each time interval. A serial correction step is applied by calculating

$$U_{n+1}^{K+1} = \mathcal{G}(t_{n+1}, t_n, U_n^{K+1}) + \mathcal{F}(t_{n+1}, t_n, U_n^K) - \mathcal{G}(t_{n+1}, t_n, U_n^K) \quad (3)$$

and propagating the result to the next processor in line. Note that the second and third terms on the right-hand side of this expression have been obtained in previous iterations and steps. Convergence is checked through the condition $\mid U_n^K - U_n^{K-1} \mid < \varepsilon$ where ε is the predetermined tolerance value. If the difference between the solutions for two successive parareal K iterations is smaller than ε for all time intervals, the parareal cycle completes ([30], [31].

We use a pipelined approach in which the fine solver begins as soon as the coarse approximation is available for the time interval to be calculated by each core, instead of waiting for all of the cores in the individual communicator to complete the $\mathcal{G}$ calculation [17], [32]. This ensures that each step is completed as quickly as possible and further reduces

the time-to-solution by removing the need for processors to remain idle while waiting for all of the other processors to finish calculating $\mathcal{G}$. It should be noted that this scheme is only advantageous when convergence occurs faster than a serial run of the fine iterator would take [21].

V. Multilevel Spatio-Temporal Coupling

To optimally use the available hardware of massively parallel supercomputers and ultimately minimize the runtime of the applications in question, we posit that leveraging both spatial and temporal domain decomposition is needed for fixed-size problems. We introduce a general approach for coupling these decomposition strategies through a multi-level spatio-temporal coupling (MSTC) scheme. This is similar to communicator breakdown introduced by Grinberg *et al.* in [8] to enable the coupling between physical models. The key advantage of the MSTC architecture is the hierarchical decomposition of the default World communicator into sub-communicators to enable efficient coupling of parallel decomposition in both time and space. This decomposition is handled by splitting the World communicator into N different sub communicators to handle the different time intervals of equal size. This is handled in a topology aware manner assigning cores that are physically near each other to the same Tier 2 ($T2$) group of cores. If no temporal decomposition is being used, all cores are assigned to the same $T2$ group. These groups are further subdivided through spatial decomposition strategies defining Tier 3, $T3$, non-overlapping groups. The core kernel being modeled in this framework would be solved within $T3$ groups.

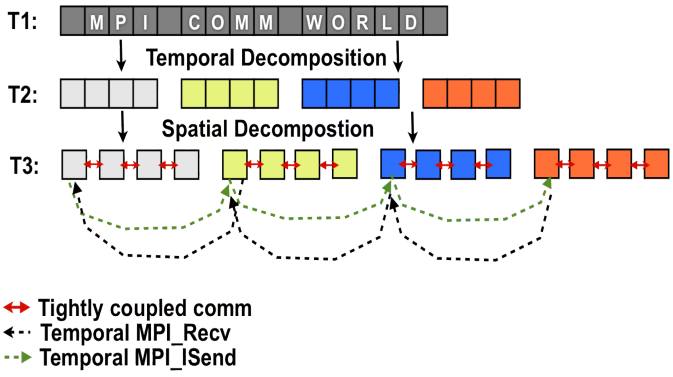

Figure 3. Multi-level Spatio-Temporal Interface breakdown. For Tier 2, the World communicator is broken into temporal into separate communicators handling temporal intervals. For Tier 3, each $T2$ group is broken up spatially. Coarse and fine solvers run across $T3$ groups. The red arrows indicate the tightly coupled message passing with the LBM and the dashed lines indicate the communication between $T2$ groups for the one core of each.

Fig. 3 shows the layout of the MSTC. At $T2$, new communicators are introduced splitting the cores into groups handling each time interval and at $T3$, these are each further decomposed to handle specific spatial regions. The arrows in $T3$ give a view of the communication patterns involved with MSTC. The red bi-directional arrows indicate the tightly coupled interaction between cores in the same $T3$ group. This is defined by the kernel in question and can involve either point-to-point or global communication across all cores within that group. The kernel here defines both the coarse and fine solvers that will be executed within each $T3$ group. The dashed arrows indicate the point-to-point communication between $T2$ groups. These provide the interaction between temporal intervals in which the core handling a set spatial region for a time interval will inform the estimation of the corresponding core that handles the same spatial region in the next time interval. This is implemented through point-to-point communication between cores that have the same $T3$ ranks. The communication shown here simply indicates messages for cores of rank 0 in each, but similar message patterns occur for each rank. The bulk of the computational time for the simulation actually occurs within the $T3$ groups themselves and the message passing between $T2$ groups only occurs N times. As the calculations in each $T3$ group need to proceed in sync, blocking receive protocols are leveraged to ensure this. The pipelined nature of the parareal algorithm though allows the use of non-blocking sends to minimize communication overhead.

The $T3$ groups are implemented as input variables to the coarse and fine solvers, so one could envision future methods where we would redefine the communicators throughout the coarse of the simulation allowing processors that would remain idle from K=0 time interval T communicators re-defining the late communicators, joining them and providing further spatial scaling (up to the limit of coarse).

VI. Space-time Parallelism for the LBM

To define our coarse and fine solvers, we use a two-level hierarchical refinement of coarse and fine grids in which a coarse grid covers the entire spatial domain and a finer grid is superimposed [33]. The coarse iterator models fluid moving using the coarse grid and similarly the fine iterator is solved across the fine grid. Unlike traditional mesh refinement methods, we cover the entire spatial domain with both the coarse grid and an overlapping superimposed finer grid. Each iterator models the fluid on its corresponding grid points separately for the entire spatial domain. To ensure accuracy and minimize communication overhead, we map the coarse and fine grid points for the same spatial region to the same core.

In the LBM, the resolution of the imposed grid spacing determines the time step size and contributes to the calculation of the kinematic viscosity of the fluid. The size of a time step, dt, is related directly to the square of the grid resolution, dx in the LBM: $dt = dx^2(\nu/\nu_o)$ where ν is the viscosity of the fluid in dimensionless lattice Boltzmann units and ν_o is the viscosity in terms of physical units

(m^2/s). The kinematic viscosity of the fluid is determined as $v = (2/\omega - 1)dx \cdot (c/6)$ where dx is the lattice spacing and $c = dx/dt$. It is the impact on these two quantities that makes the adaptation of parareal for the LBM a challenging task and not a straightforward implementation. The coupling for the fluid model on the coarse and fine grid introduces potential accuracy and stability issues to the method that need to be addressed.

As in our initial studies applying simply temporal decomposition to laminar flow in a tube [34], we use a local second-order refinement solution for coupling between the grids that relies on different relaxation parameters and lattice spacing to transition between the grids. The relaxation parameter ω in Eq. (1) must be rescaled to keep the viscosity constant across both the coarse and fine grids to ensure a stable coupling mechanism [33]. We redefine ω as:

$$\omega_f = \frac{\Delta x_c}{\Delta x_f}\left(\omega_c - \frac{1}{2}\right) + \frac{1}{2},$$

where Δx_c and Δx_f are the spatial discretization size for the coarse and fine grids, respectively, and ω_c and ω_f are the corresponding relaxation parameters [35]. This both addresses the stability concern and imposes an upper bound on N by requiring ω_c to remain close to 2 and ω_f to be greater than 1 [33]. Moreover, the use of this modified definition of ω for each grid imposes a finite limit on the disparity between the two iterator's resolutions.

Building on this we can then adapt the MSTC to this locally embedded version of the LBM using the following steps:

0. $(K = 0)$
   Initialize $T2$ and $T3$ groups.
   Define neighbors between $T2$ groups.
   In each $T3$ group, initialize the coarse estimation with serial LBM simulation for the time domain $[t_{n-1}, t_n]$ for the assigned spatial region.
   Interpolate to initialize the fine solver.

   For $(K = K + 1)$
1) $\mathcal{F}$ is across all $T3$ groups starting with the initial values provided by the previous iteration for t_{n-1} to determine the distribution function at t_n for its respective interval of time and space.
2) In each $T3$, $\mathcal{G}$ is applied. The correction to $\mathcal{F}$ is calculated via Eq. (3).
3) This result is coarsened and propagated between $T2$ groups to update the initial conditions for $\mathcal{G}$.
4) Convergence is checked: if all intervals of time have converged, exit the cycle; else, return to Step 1.

The key components to this scheme are the steps required to link the two different grid resolution levels in the *coarsening* and the *interpolation* steps. On each core, a coarsening function in which the distribution function at each velocity for the fine grid is averaged and rescaled is required to move data between the two grid levels. This operates on conserved values and introduces no further truncation error. Similarly, an interpolation method is required to address the new lattice sites required by the fine iterator. It is in both of these functions that the rescaling of ω is employed to maintain a constant kinematic viscosity across the spatio-temporal decompositions.

To enable the pipeline approach in a spatio-temporal coupling, we employ non-blocking sending of messages but have the receivers leverage blocking calls. This prevents cores from within one sub communicator to get out of sync. All cores in the sub communicator must handle the advection and collision within one step sequentially as the following step relies on data from its nearest neighbors from the previous step.

It is worth noting that even as larger fluid systems are modeled, the overhead of the message passing between $T2$ groups will remain constant. This is due to the fact that the gain from spatial speedup should always be maximized before using temporal decomposition. In the case of lattice Boltzmann, this means that temporal decomposition will only be employed when the number of fluid nodes per core drops below the cutoff defined by Fig. 1. Adhering to this drop off imposes a fundamental limit to the potential number of fluid nodes to be handles on each core within a $T2$ group and subsequently a limit to the potential message size being sent between K iterations.

A. Speedup Calculation

To obtain a quantitative understanding of the potential performance improvement to be gained from applying this spatio-temporal parallelization technique to the LBM, we must assess the upper bound on the method's strong scaling capabilities. Here we assume that the cost of communication between processors is negligible and that the cores are homogenous. We define the computational cost for the fine solver as the cost per time step multiplied by the number of time steps in one K iteration, denoted by γ_F. Similarly, we use γ_G to indicate the cost of the coarse solver. Following the procedure outlined by Minion *et al.*, we calculate the parallel speedup, S, for pipelined parareal calculations using

$$S = \frac{N\gamma_F}{N\gamma_G + K(\gamma_G + \gamma_F)} = \frac{1}{\alpha + \frac{K}{N}(\alpha + 1)} \qquad (4)$$

with N the total number of cores and K the number of parareal iterations, and we have defined $\alpha = \gamma_G/\gamma_F$. This model was used to demonstrate the speedup at different K iterations on 32,768 cores in Fig. 7.

Eq. 4 allows the estimation of speedup achieved by simply the temporal component of the space-time coupling. It can be taken alongside the speedup from spatial scaling to provide the predicated total speedup. Extending the model itself to calculate the combined speedup, we need to define S_F and S_G as the parallel speedup of the spatial decomposition on

P cores for each respective iterator. In this instance, the total number of cores in the system is $N * P$ where each N time interval is divided into P spatial domains. In the work presented here, the resolution of the fine grid is twice that of the coarse grid. This variance corresponds to the fine iterator having four times as many time steps as the coarse. This imposes a limit of 32 to the cost ratio of the coarse to fine iterators. This limit appears in Eq. 5 defining the speedup from the spatio-temporal scheme.

$$S_{MSTC} = \frac{1}{\frac{S_F}{32S_G} + \frac{K}{N}(\frac{S_F}{32S_G} + 1)} \tag{5}$$

In both temporal and spatio-temporal decomposition, the potential speedup can be optimized through the minimization of $\frac{K}{N}$.

VII. Numerical Results

In this section, we present results of experiments on the accuracy and speedup obtained from the application of the MSTC method to modeling hemodynamic properties in a patient suffering from co-arctation of the aorta (CoA). Personalized computer simulations can provide an insightful study of the flow under stress conditions that would otherwise require difficult stress tests that have potential side effects. In the following studies, we use patient data from an 8-year old female with moderate aortic co-arctation (65% area reduction). Gadolinium-enhanced MR angiography was performed using a 1.5-T GE Sigma scanner to obtain the arterial geometry as shown in Fig 5 (a). We assume rigid walls and Newtonian flow behavior for the blood, with a density r = 0.001 gr/mm3 and a dynamic viscosity m = 0.004 gr/mm/sec [36]. All of these studies were completed using an IBM Blue Gene/Q supercomputer.

The simulation is setup with a 100 μm resolution Cartesian grid for the coarse iterator and a 50 μm resolution grid for the fine iterator. The fine grid corresponds to the fluid system size matching the red line in Fig. 1 allowing us to focus on reducing the time-to-solution for simulations in which adding more cores to a spatial parallelization will no longer improve the parallel performance. Unless otherwise noted, the following simulations were conducted on 65,536 cores of the IBM Blue Gene/Q supercomputer. We selected $N = 8$ as the number of temporal domains so that we would be maximizing the strong scaling potential for this fluid system. By using 8 time intervals, each sub communicator consists of 8,192 cores. The time duration simulated was 0.7 seconds or the average length of one human heartbeat. The goal of this work was to shorten the overall time-to-solution, so we focus on the strong scaling capabilities in which a fixed system size is used as we increase the number of processors.

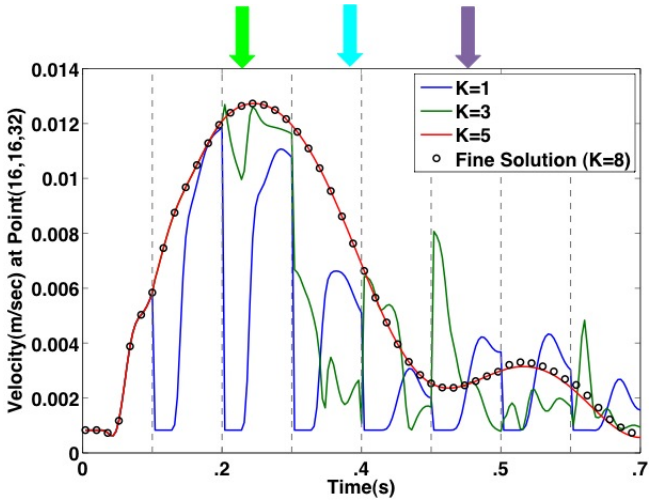

Figure 4. Pulsatile Flow. Test to recover time dependent phenomena for a system broken into $N = 8$ temporal domains simulated on 65,536 cores. The blue line shows the magnitude of the velocity over time at point (16,16,32) after the first K iteration. The green line and red line represent $K = 3$ and $K = 5$ respectively. The black circles indicate the result of the fine solver which is equivalent to $K = 8$. The vertical dashed lines indicate the break point between regions of time handled by each core. The block arrows indicate the time points at which the accuracy is later assessed across the aorta in Fig. 5.

A. Time Dependent Phenomena

In order to determine if use of MSTC could accurately recover time dependent phenomena, we introduced pulsatile flow via the Zou-He boundary conditions [37]. To this end, a patient-specific inflow velocity was prescribed at the inlet and a constant pressure gradient was applied out the outlets. The inflow velocity was obtained via a 2D, phase-contrast (PC) MRI sequence with through-plane velocity encoding [36]. In order to enable continuous flow throughout the heartbeat, we use a sum of sine functions to the data to determine the equation of the pulse. The fit procedure was performed in MATLAB using a non-linear least squares method and a trust region algorithm [38]. Using a Pearson correlation, there was a statistically significant agreement between the phase contrast data and the equation derived velocity values ($r = 0.981, p < 7 * 10^{-15}$). Fig. 4 shows that the spatio-temporal framework in HARVEY starts to recover the pulsatile behavior with greater accuracy at each K iteration.The magnitude of the fluid velocity at point (16,16,32) for a range of K levels of a simulation with 8 temporal domain slices using 65,536 cores is presented. As the K iteration level increases, the result gets nearer and nearer to the solution of the full fine solver which is equivalent to the $K = 8$ depicted by the black circles. Even at $K = 5$, the time dependent behavior is fully recovered, as shown by the red lines. The dashed black vertical lines indicate the break point between regions of time handled by different time intervals or sub communicators.

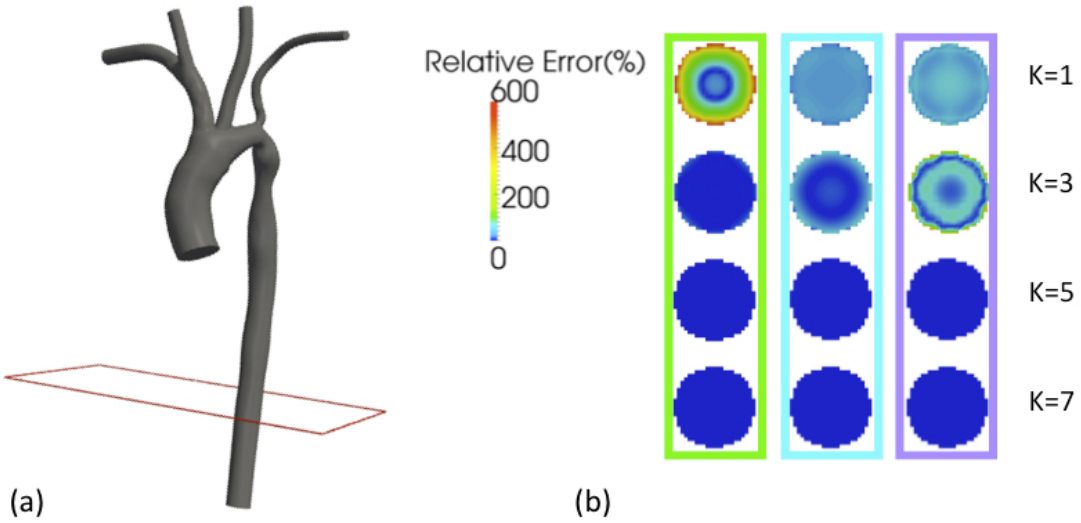

Figure 5. Accuracy at different K levels. (a) The mesh defining the arterial geometry from patient specific data is shown. The red rectangle depicts the section across which velocity is assessed. (b) The three vertical rectangles correspond to the time points marked in Fig. 4 and identify the time points that the error tests were imposed over the coarse of one heartbeat. The relative error in velocity as compared to the solution of the fine iterator, $\mathcal{F}$, is shown at four different K levels at each time point identified in 4. The error variation across the section is highlighted.

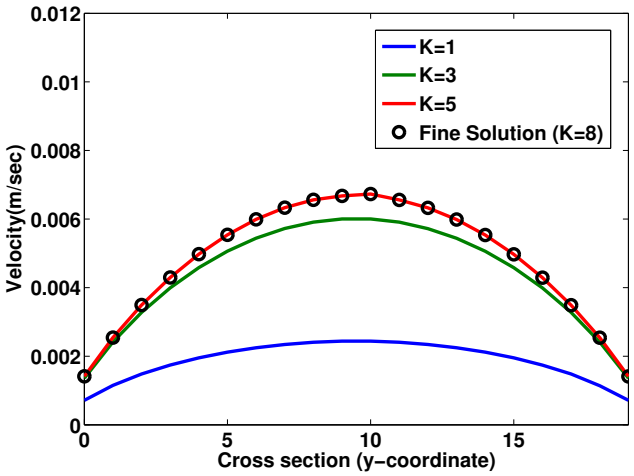

Figure 6. Accuracy test for a patient model broken into $N = 8$ temporal domains simulated on 32,768 cores.

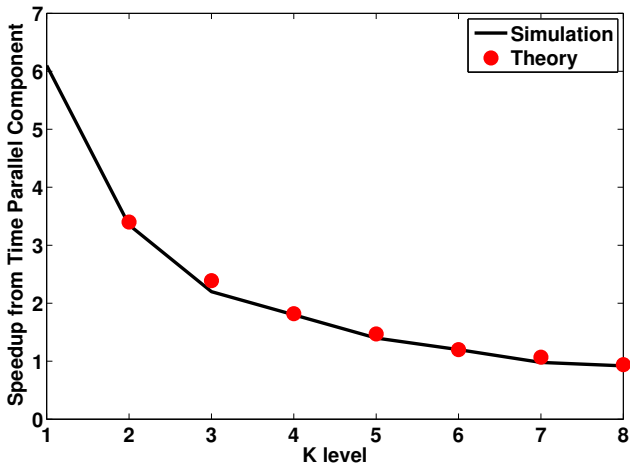

Figure 7. Performance tests demonstrating the strong correlation between the theoretically expected performance and experimental results. The black line depicts the simulation results and the red circles indicate the theoretical speedup added from the temporal component as calculated from Eq. (4).

B. Accuracy

Full convergence with machine accuracy to the $\mathcal{F}$ solution defined in Section IV requires $K = N$ iterations when using N processors. The results presented and discussed in this section are intended to show that convergence within a set tolerance can be achieved with fewer K iterations than the number of processors ($K < N$). Fig. 5 shows the change in accuracy across the slice at the red plane within a real patient's arterial geometry as shown in Fig. 5(a). The relative error in velocity as compared to the solution of the fine iterator, $\mathcal{F}$, is shown at three different time points from within a heartbeat. The pulsatile nature of the flow causes a variation in the error. At each K iteration, the overall accuracy increases. When $K > 5$, the relative error across the entire slice is approximately zero. The fact that the initial time intervals reach convergence first is highlighted by $K = 3$ in which the error is greatly reduced for the first two time points. Of further note, the greatest error is at the center of the tube. The calculated velocity at the wall of the

tube converges to the result of the fine iterator at a faster rate. This is significant in selecting the desired K level as it can depend on the research question and disease targeted. Often when assessing risk for a disease like atherosclerosis, one is concerned with the magnitude of the endothelial shear stress on the wall of the vessel [1]. In this case a lower K iteration may provide the accuracy desired. Conversely, when trying to understand the pressure gradient associated with co-arctation of the aorta, the pressure at the center of the vessel is equally as significant.

These results are congruent with results found for other domains. Baffico *et al.* showed that $K = 4$ of a domain broken into six temporal intervals provided accurate results for a molecular dynamics code [14] and Fisher *et al.* demonstrated high accuracy at $K = 2$ for a Navier-Stokes simulation with ten temporal intervals [21].

We then assessed the additional speedup provided by our method above and beyond that achieved by the spatial parallelization and investigated how close our implementation comes to meeting the theoretical performance prescribed by Eq. (5). Fig. 7 shows the correlation between the theoretical performance model previously discussed for speedup, Eq. (4), and the experimental results of the simulation. The previously mentioned resolutions ($\Delta x_f = 50$ μm and $\Delta x_c = 100$ μm resolution) were used to determine the value of α and consequently the overall computational costs. Any change to either grid resolution would impact the associated speedup that can be achieved.

C. Time-to-Solution

We evaluate the performance of our implementation of our MSTC scheme with LBM on a 65,536-core system. Our approach sees a reduction in the time-to-solution at each K level (Fig. 8). The black line depicts the results from simply using spatial decomposition while the red and green lines show the runtime using $K = 5$ and $K = 3$ respectively. While the minimal runtime for the spatial decomposition is achieved with only 4,096 processors, the spatio-temporal decomposition extends the scalability limit and provides even lower runtimes at 65,536 cores. For example, the result at 65,536 cores for $K = 5$ demonstrates a 40% reduction in runtime as compared to the minimal time achieved through spatial decomposition alone.

Table I shows the relative error in velocity at both the wall of the vessel and the center. The significance of this data is that we can more efficiently use available hardware by combining the use of temporal and spatial scaling. As the data shows, for all iterations with $K \geq 5$, the relative error is less than 1%.

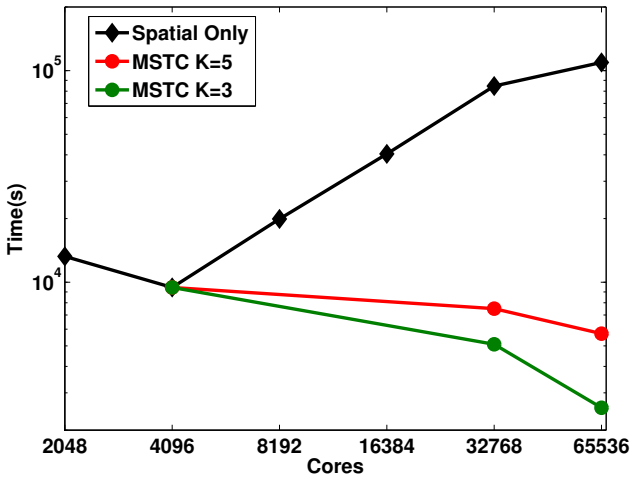

Figure 8. Time-to-solution for different levels of K when using MSTC as compared to a simulation using only spatial decomposition.

Table I
PERCENT ERROR

K level	At Wall	At Vessel Center
1	401%	823%
2	56%	225%
3	9%	31%
4	2%	5%
5	<1%	<1%
6	<1%	<1%
7	<1%	<1%
8	<1%	<1%

VIII. CONCLUSIONS

For many fluid problems even beyond the medical applications discusses in this paper, there is a strong need to model longer time durations for fixed system sizes. In these instances, there is often a fundamental limit to the benefits that can be obtained through conventional spatial scaling. Through the careful coupling of temporal with spatial parallelization, we have shown that we efficiently use available parallel resources to reduce the time-to-solution for real problems and real data. As wallclock time plays a crucial role in the potential impact of these methods, the results presented above show that spatio-temporal parallelism is has the potential to extend scaling possibilities beyond the scaling limit imposed by traditional strong scaling. We have demonstrated that parallelizing the lattice Boltzmann method in both time and space is a successful method for time dependent blood flow in the arterial system. Using patient specific data reconstructed from Magnetic Resonance Angiography, we have recovered steady and pulsatile flow fields within an acceptable accuracy.

We showed that implementing the MSTC method in HARVEY resulted in a 40% reduction in the overall wallclock time for the simulation of a real problem. This minimization of the overall time-to-solution was achieved using software approaches that exploited the hardware's low latency communication mechanisms and tight, fine-grained coupling between grid levels. As a result, the coupling method presented in this paper allows researchers

to overcome the fundamental strong scaling limit imposed by spatial scaling alone. We believe that use of the MSTC mechanism will enable simulations of longer physical time periods in shorter wallclock times, enabling the study of phenomena outside the reach of traditional CFD methods. Moreover, the techniques presented in this paper are agnostic to the scaling limit itself in the sense that they would continue to add the same speedup factor above the spatial limit no matter where the spatial limit is place. This is allows the MSTC scheme to continuously extend the potential of new developments improving the spatial scaling limit. As systems with millions of cores become more prevalent, these methods provide a new way for more codes that traditionally could not make use of the full system to effectively scale to large core count and enable science of unprecedented scale.

IX. Acknowledgements

This work was supported by the Department of Energy's Computational Science Graduate Fellowship under grant number DOE CSGF DE-FG02-97ER25308 and in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy award DE-SC0004131. We would like to thank Franziska Michor, the Dana Farber Cancer Institute, and Erik Draeger, Lawrence Livermore National Laboratory, for useful discussions regarding this research. This is LLNL report LLNL-PROC-644182.

References

[1] S. Melchionna, M. Bernaschi, S. Succi, E. Kaxiras, F. J. Rybicki, D. Mitsouras, A. U. Coskun, and C. L. Feldman, "Hydrokinetic approach to large-scale cardiovascular blood flow," *Computer Physics Communications*, vol. 181, no. 3, pp. 462–472, 2010.

[2] e. a. Koo, B-K, "Diagnosis of ischemia-causing stenoses obtained via non-invasive fractional flow reserve (discoverflow): a prospective multicentre first-in-man study." in *Proceedings of EuroPCR: the annual meeting of the European Association for Percutaneous Cardiovascular Interventions.*, ser. PCR '11, 2011.

[3] C. A. Taylor, M. T. Draney, J. P. Ku, D. Parker, B. N. Steele, K. Wang, and C. K. Zarins, "Predictive medicine: computational techniques in therapeutic decision-making," *Computer Aided Surgery*, vol. 4, no. 5, pp. 231–247, 1999.

[4] C. A. Taylor, T. J. Hughes, and C. K. Zarins, "Finite element modeling of blood flow in arteries," *Computer methods in applied mechanics and engineering*, vol. 158, no. 1, pp. 155–196, 1998.

[5] D. A. Vorp, D. A. Steinman, and C. R. Ethier, "Computational modeling of arterial biomechanics," *Computing in Science & Engineering*, vol. 3, no. 5, pp. 51–64, 2001.

[6] L. Grinberg, T. Anor, J. Madsen, A. Yakhot, and G. Karniadakis, "Large-scale simulation of the human arterial tree," *Clinical and Experimental Pharmacology and Physiology*, vol. 36, no. 2, pp. 194–205, 2009.

[7] A. Peters, S. Melchionna, E. Kaxiras, J. Lätt, J. Sircar, M. Bernaschi, M. Bison, and S. Succi, "Multiscale simulation of cardiovascular flows on the IBM Blue Gene/P: Full heart-circulation system at red-blood cell resolution," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '10. IEEE Computer Society, 2010.

[8] L. Grinberg, V. Morozov, D. Fedosov, J. A. Insley, M. E. Papka, K. Kumaran, and G. E. Karniadakis, "A new computational paradigm in multiscale simulations: Application to brain blood flow," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*. IEEE, 2011, pp. 1–12.

[9] J. Clausen, D. Reasor Jr, and C. Aidun, "Parallel performance of a lattice-boltzmann/finite element cellular blood flow solver on the ibm blue gene/p architecture," *Computer Physics Communications*, vol. 181, no. 6, pp. 1013–1020, 2010.

[10] C. Körner, T. Pohl, U. Rüde, N. Thürey, and T. Zeiser, "Parallel lattice boltzmann methods for cfd applications," in *Numerical Solution of Partial Differential Equations on Parallel Computers*. Springer, 2006, pp. 439–466.

[11] A. Peters Randles, V. Kale, J.R. Hammond, W. Gropp, and E. Kaxiras, "Performance analysis of the lattice Boltzmann model beyond Navier-Stokes," in *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium*, ser. IPDPS '13, 2013.

[12] J. Lions, Y. Maday, and G. Turinici, "A parareal in time discretization of PDE's," *C.R. Acad. Sci. Paris, Serie I*, vol. 332, pp. 661–668, 2001.

[13] G. Bal and Y. Maday, "A parareal time discretization for non-linear pdes with application to the pricing of an american put," in *Recent developments in domain decomposition methods*. Springer, 2002, pp. 189–202.

[14] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, "Parallel-in-time molecular-dynamics simulations," *Phys. Rev. E*, vol. 66, p. 057701, Nov 2002.

[15] C. Farhat and M. Chandesris, "Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications," *International Journal for Numerical Methods in Engineering*, vol. 58, no. 9, pp. 1397–1434, 2003.

[16] C. Farhat, J. Cortial, C. Dastillung, and H. Bavestrello, "Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses," *International journal for numerical methods in engineering*, vol. 67, no. 5, pp. 697–724, 2006.

[17] M. Minion, "A hybrid parareal spectral deferred corrections method," *Communications in Applied Mathematics and Computational Science*, vol. 5, pp. 265–301, 2010.

[18] I. P. Gent, C. Jefferson, and I. Miguel, "Minion: A fast scalable constraint solver," *Frontiers in Artificial Intelligence and Applications*, vol. 141, p. 98, 2006.

[19] 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*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 92:1–92:11.

[20] J. Reynolds-Barredo, R. Sanchez, and L. Berry, "Modelling parareal convergence in 2D drift wave plasma turbulence," in *High Performance Computing and Simulation (HPCS), 2012 International Conference on*. IEEE, 2012, pp. 726–727.

[21] P. Fischer, F. Hecht, and Y. Maday, "A parareal in time semi-implicit approximation of the Navier-Stokes equations," in *Proceedings of Fifteen International Conference on Domain Decomposition Methods*. Springer Verlag, 2004, pp. 433–440.

[22] D. P. Zipes and H. J. Wellens, "Sudden cardiac death," *Circulation*, vol. 98, no. 21, pp. 2334–2351, 1998.

[23] F. Higuera and J. Jimenez, "Boltzmann approach to lattice gas simulations," *EPL (Europhysics Letters)*, vol. 9, no. 7, p. 663, 1989.

[24] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, 2001.

[25] J. Carter, M. Soe, L. Oliker, Y. Tsuda, G. Vahala, L. Vahala, and A. Macnab, "Magnetohydrodynamic turbulence simulations on the earth simulator using the lattice Boltzmann method," in *Proceedings of the 2005 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '05. IEEE Computer Society, 2005.

[26] S. Williams, L. Oliker, J. Carter, and J. Shalf, "Extracting ultra-scale lattice Boltzmann performance via hierarchical and distributed auto-tuning," in *Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. IEEE Computer Society, 2011, pp. 1–10.

[27] T. Pohl, F. Deserno, N. Thurey, U. Rude, P. Lammers, G. Wellein, and T. Zeiser, "Performance evaluation of parallel large-scale lattice Boltzmann applications on three supercomputing architectures," in *Proceedings of the 2004 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '04. IEEE Computer Society, 2004.

[28] N. S. Martys and H. Chen, "Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice boltzmann method," *Physical Review E*, vol. 53, no. 1, p. 743, 1996.

[29] P. Bhatnagar, E. Gross, and M. Krook, "A model for collision processes in gases," *Physics Review Letters*, vol. 94, p. 511, 1954.

[30] J. Reynolds-Barredo, D. Newman, R. Sanchez, D. Samaddar, L. Berry, and W. Elwasif, "Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations," *Journal of Computational Physics*, 2012.

[31] D. Samaddar, D. Newman, and R. Sánchez, "Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm," *Journal of Computational Physics*, vol. 229, no. 18, pp. 6558–6573, 2010.

[32] N. Kok Fu and N. Mohd Ali, "Improving pipelined time stepping algorithm for distributed memory multicomputers," *Sains Malaysiana*, vol. 39, no. 6, pp. 1041–1048, 2010.

[33] O. Filippova and D. Hänel, "Grid refinement for lattice-BGK models," *Journal of Computational Physics*, vol. 147, no. 1, pp. 219–228, 1998.

[34] A. Peters Randles and E. Kaxiras, "Parallel in time approximation of the lattice Boltzmann method for laminar flows," March 2013, (submitted).

[35] A. Dupuis and B. Chopard, "Theory and applications of an alternative lattice Boltzmann grid refinement algorithm," *Physical Review E*, vol. 67, no. 6, p. 066707, 2003.

[36] J. F. Ladisa, A. Figueroa, I. E. Vignon-Clementel, H. Jin Kim, N. Xiao, L. M. Ellwein, F. P. Chan, J. A. Feinstein, C. A. Taylor *et al.*, "Computational simulations for aortic coarctation: representative results from a sampling of patients." *Journal of biomechanical engineering*, vol. 133, no. 9, p. 091008, 2011.

[37] Q. Zou and X. He, "On pressure and velocity boundary conditions for the lattice boltzmann bgk model," *Physics of Fluids*, vol. 9, p. 1591, 1997.

[38] MATLAB, *version 8.0.0.783 (R2012b)*. Natick, Massachusetts: The MathWorks Inc., 2012.