Numerical Simulation of Three-dimensional Blood Flows Using Domain Decomposition Method on Parallel Computer[☆]

Feng-Nan Hwang[a], Chao-Ying Wu[a], Xiao-Chuan Cai[b]

[a] *Department of Mathematics, National Central University, Jhongli, 32001, Taiwan*
[b] *Department of Computer Science, University of Colorado, Boulder, CO 80309, USA*

Abstract

A good numerical blood flow simulation tool based on patient-specific anatomy and physiological conditions can be clinically helpful for physicians or researchers to study vascular diseases, to enhance diagnoses, as well as to plan surgery procedures. Such a tool is computationally very expensive, and often requires the use of large scale supercomputers with many core processors. In this paper, we focus on developing parallel domain decomposition algorithms for solving nonlinear systems arising from the discretization of three-dimensional blood flow model equations with a stabilized finite element method for the spatial variables and an implicit backward Euler finite difference method for the temporal variable. More precisely speaking, at each time step, the resulting nonlinear system is solved by the Newton-Krylov-Schwarz algorithm. We implement the parallel fluid solver using PETSc and integrate it with other state-of-the-art software packages into a parallel blood flow simulation system, which includes Cubit, ParMETIS and ParaView for mesh generation, mesh partitioning, and visualization, respectively. We validate our parallel code and investigate the parallel performance of our algorithms for both a straight artery model and an end-to-side graft model.

Key words: Domain decomposition methods, blood flow modeling, parallel processing, Newton-Krylov-Schwarz algorithm

[☆]The first two authors were also supported in part by the National Science Council of Taiwan, 96-2115-M-008-007-MY2. The last author was supported in part by the Department of Energy, DE-FC02-01ER25479, and in part by the US National Science Foundation, CCR-0219190, ACI-0072089 and ACI-0305666

Email addresses: hwangf@math.ncu.edu.tw (Feng-Nan Hwang), cai@cs.colorado.edu (Xiao-Chuan Cai)

1. Introduction

In recent years there is growing interest in parallel computational techniques for the study of blood flow dynamics in vascular systems [3, 4, 21, 28, 32]. Numerous computational models have been developed to describe the local blood flow field and to simulate the response of the vessel walls under certain hemodynamic conditions [24, 30, 31]. The blood characteristics of interests are often shown as abnormal local flow structures such as vortices, flow separations, recirculation and stagnation areas; blood pressure drop; high shear stress regions and system total energy dissipation [14, 15]. These characteristics play critical roles in the formation of serious vascular diseases such as atherosclerotic lesions and thrombus formation. The ultimate goal of the computational biofluid research is to provide reliable tools to illustrate the hemodynamic properties under specific conditions and to predict their changes when certain disturbances arise. Good simulation results can be used clinically to help physicians to understand vascular diseases.

In this research, we develop a scientific software system based on parallel domain decomposition methods for simulating complex biofluid problems according to patient-specific vessel geometries and flow conditions. A finite element method with unstructured meshes is suitable to handle the complicated geometries of the blood flow problem [2, 9, 11, 25]. In general, finite element-based scientific simulations comprise three basic steps [18, 25]: (1) preprocessing step, (2) solution step, and (3) postprocessing step. During the preprocessing step, one first uses an image-to-surface definition tool to construct a geometric solid model of a patient's arteries based the images obtained from some medical imaging devices, such as the Nuclear Magnetic Resonance, the Computed Tomography or the Digital Substraction Angiography, then generates an unstructured 3D finite element mesh for the complex geometries of the arteries. On the other hand, some non-invasive device, like Doppler Ultrasound or Phase-Contrast Magnetic Resonance is used to measure some of the flow condition, such as the velocity profiles and pressure drop at some point. This collected data is useful for setting of the realistic boundary conditions for the blood flow simulation. The solution processing step requires a robust and efficient time-dependent flow solver to handle the high nonlinearity and complex geometry of the blood flow problem. To resolve the detailed changes of the hemodynamic equations in complex and delicate vascular systems through numerical simulation, the use of highly refined unstructured finite element meshes is essential. This implies that the resulting ill-conditioned large algebraic systems need to be solved. Although there are commercial

fluid software packages available, most of them are sequential. The large amount of required computational time makes some simulations infeasible. Consequently, a fast parallel computational algorithm and high performance software are needed to overcome such limitations.

In this paper, we focus on developing parallel, scalable, robust domain decomposition algorithms [22, 27, 29] for solving linear and nonlinear systems arising from the discretization of blood flow model equations, where a stabilized finite element method is used for the spatial discretization and a fully implicit backward Euler integrator for the temporal discretization. At each time step, the Newton-Krylov-Schwarz (NKS) algorithm [5, 16] is employed for solving resulting nonlinear algebraic equations. The NKS algorithm consists of three important components. The inexact Newton method with backtracking (INB) [6, 20] is used as a nonlinear solver. In each Newton step, an overlapping Schwarz-type preconditioner is employed to accelerate the convergence of a Krylov subspace methods, which is used for solving linear Jacobian systems. The Schwarz-type preconditioner is suitable for parallel computing, since all subdomain problems are independent of each other and can be solved in parallel. In addition, we integrate our parallel time-dependent fluid solver implemented by using Portable, Extensible Toolkit for Scientific Computation (PETSc) [1] with other state of the art parallel software packages into a parallel blood flow simulation system. These packages include (1) CUBIT [34] for unstructured finite element mesh generation; (2) ParMETIS [13] for mesh partition; (3) ParaView [35] for displaying numerical results and conducting data analysis.

The rest of this paper is organized as follows. In the next section, we briefly mention the blood flow model based on incompressible Navier-Stokes equations. In Section 3, we provide a description of a Newton-Krylov nonlinear solver, a parallel Schwarz preconditioner for the saddle-point type Jacobian system. An overview of the parallel blood flow simulation system is outlined in Section 4. In Section 5, numerical results on two blood flow problems are presented. Concluding remarks are given in Section 6.

2. Blood flow model and its spatial and temporal discretizations

In this study, we treat the wall of arteries as rigid and fixed. This assumption is acceptable in some situation when the diameter of the arteries is varied much less than 5-10% during a cardiac cycle in the case of diseased vessels in which we are particularly interested. We also assume blood

flows to be incompressible and laminar [7, 23]. Under these assumptions, blood flows in large arteries can be modeled by 3D unsteady incompressible Newtonian flow equations defined on a bounded domain Ω with the boundary $\Gamma = \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_{wall}$, which are given by

$$\begin{cases} \rho\left(\dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) + \nabla\cdot\sigma = 0 & \text{in} \quad \Omega\times(0,T), \\ \nabla\cdot\boldsymbol{u} = 0 & \text{in} \quad \Omega\times(0,T), \\ \boldsymbol{u} = 0 & \text{on} \quad \Gamma_{wall}\times(0,T), \\ \boldsymbol{u} = g & \text{on} \quad \Gamma_{in}\times(0,T), \\ \sigma\cdot\boldsymbol{n} = \boldsymbol{0} & \text{on} \quad \Gamma_{out}\times(0,T), \\ \boldsymbol{u} = \boldsymbol{u}_0 & \text{in} \quad \Omega \text{ at } t = 0, \end{cases} \tag{1}$$

where $\boldsymbol{u} = (u_1, u_2, u_3)^T$ is the velocity, ρ is the blood density. Here, we impose a periodic parabolic type profile, for example, the solution of Womersley flow, on the inflow boundary, Γ_{in}, the no-slip boundary condition on the wall, Γ_{wall}, and the Neumann type boundary condition on the outflow boundary, Γ_{out}. For $\boldsymbol{u}_0$, we assume that the flow is steady at the beginning of computation. For viscous incompressible fluids, the Cauchy stress tensor can be decomposed into two parts:

$$\sigma = -p\boldsymbol{I} + 2\mu\boldsymbol{D},$$

where p is the hydrostatic pressure, $\boldsymbol{I}$ is a second-order identity tensor, μ is the dynamic viscosity, and $\boldsymbol{D}$ is a second-order deformation rate tensor defined as $\boldsymbol{D} = \frac{1}{2}[(\nabla\boldsymbol{u}) + (\nabla\boldsymbol{u})^T]$. The Reynolds number is defined as $Re = \frac{\rho\overline{V}D}{\mu}$, where $\overline{V}$ is the characteristic velocity and D is the characteristic length. Additionally, the Womersley number, which plays an important role in biofluid mechanics, is defined as $\alpha = \sqrt{\frac{\rho\omega}{\mu}}D$ and represents the ratio of the unsteady inertia force to the viscous force. Here, ω is the angular frequency of the oscillatory flows.

To discretize (1), we use an implicit backward Euler finite difference method in the temporal domain and a $P_1 - P_1$ stabilized finite element method [8] in the spatial domain on a given unstructured tetrahedral mesh, $\mathcal{T}^h = \{K\}$. Let V_h and P_h be a pair of finite element spaces for the velocity and pressure:

$$V_h = \{\boldsymbol{v} \in (C^0(\Omega)\cap(H^1(\Omega))^3 : \ \boldsymbol{v}|_K \in P_1(K)^3, \ K \in \mathcal{T}^h \ \}$$

and

$$P_h = \{p \in C^0(\Omega)\cap L^2(\Omega) : \ p|_K \in P_1(K), \ K \in \mathcal{T}^h\}.$$

Here $C^0(\Omega)$ is the set of all continuous functions defined on Ω, $L^2(\Omega)$, and $H^1(\Omega)$ are the standard notations with the usual meanings in the finite element literature [9, 25]. The weighting and trial velocity function spaces V_h^0 and V_h^g are

$$V_h^0 = \{\boldsymbol{v} \in V_h : \boldsymbol{v} = 0 \text{ on } \Gamma_{in} \cup \Gamma_{wall}\} \text{ and}$$

$$V_h^g = \{\boldsymbol{v}(\cdot, t) \in V_h, t \in [0, T] : \ \boldsymbol{v} = g \text{ on } \Gamma_{in} \text{ and } \boldsymbol{v} = 0 \text{ on } \Gamma_{wall}\}.$$

Similarly, P_h is used for both the weighting and trial pressure function spaces. A fully implicit stabilized finite element method takes the form: Find $\boldsymbol{u}_h^{(n+1)} \in V_h^g$ and $p_h^{(n+1)} \in P_h$, such that

$$B(\boldsymbol{u}_h^{(n+1)}, p_h^{(n+1)}; \boldsymbol{v}_h, q_h) = 0 \quad \forall (\boldsymbol{v}_h, q_h) \in V_h^0 \times P_h \tag{2}$$

with

$$B(\boldsymbol{u}, p; \boldsymbol{v}, q) = \left(\rho\left(\frac{\boldsymbol{u} - \boldsymbol{u}^{(n)}}{\Delta t}\right), \boldsymbol{v}\right) + ((\rho\nabla\boldsymbol{u})\cdot\boldsymbol{u}, \boldsymbol{v}) + (\mu\nabla\boldsymbol{u}, \nabla\boldsymbol{v}) - (\nabla\cdot\boldsymbol{v}, p) -$$
$$(\nabla\cdot\boldsymbol{u}, q) + \sum_{K\in\mathcal{T}^h} (\rho(\frac{\boldsymbol{u} - \boldsymbol{u}^{(n)}}{\Delta t}) + (\nabla\boldsymbol{u})\cdot\boldsymbol{u} + \nabla p, \tau((\nabla\boldsymbol{v})\cdot\boldsymbol{v} + \nabla q))_K + (\nabla\cdot\boldsymbol{u}, \delta\nabla\cdot\boldsymbol{v}),$$

where $\boldsymbol{u}^{(n)}$ is the velocity at the current time step, and $\boldsymbol{u}^{(n+1)}$ and $p^{(n+1)}$ are unknown velocity and pressure, respectively, at the next time step. We use the stabilization parameters τ and δ suggested in [8]. Or, equivalently, at each time step, we solve a large, sparse, nonlinear algebraic system

$$F(x^*) = 0, \tag{3}$$

where the vector x^* corresponds to the nodal values of $\boldsymbol{u}_h = (u_h^1, u_h^2, u_h^3)$ and p_h at time $t = (n+1)\Delta t$.

3. A parallel Newton-Krylov-Schwarz algorithm

To solve the large nonlinear systems (3), we employ a NKS algorithm, which can be briefly described as follows. Let the solution at the previous time step be an initial guess, $x^{(0)}$, for Newton iterations, and assume $x^{(k)}$ is the current approximation of x^*. Then a new approximation $x^{(k+1)}$ can be computed via the following steps:

Step 1: Find a Newton direction $s^{(k)}$ by solving the following preconditioned Jacobian system approximately by a Krylov subspace method, such as GMRES [26],

$$J_k M_k^{-1} y = -F(x^{(k)}), \text{ with } s^{(k)} = M_k^{-1} y, \tag{4}$$

where J_k is the Jacobian of F evaluated at $x^{(k)}$ and M_k^{-1} is called a right preconditioner. We will discuss the constructions of M_k^{-1} further in this subsection later.

Step 2: Obtain the new approximation $x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)}$, where $\lambda^{(k)} \in (0, 1]$ is a damping parameter.

In NKS, the accuracy of the solution to the Jacobian systems is controlled by the parameter, η_k, to force the condition

$$\|F(x^{(k)}) + F'(x^{(k)}) s^{(k)}\| \leq \eta_k \|F(x^{(k)})\|$$

to be satisfied. η_k is often referred to as the forcing term. If the chosen forcing term is small enough, the algorithm reduces to the exact Newton algorithm. The step length, $\lambda^{(k)} \in [\lambda_{\min}, \lambda_{\max}] \subset (0, 1)$, is selected so that

$$f(x^{(k)} + \lambda^{(k)} s^{(k)}) \leq f(x^{(k)}) + \beta \lambda^{(k)} \nabla f(x^{(k)})^T s^{(k)}, \tag{5}$$

where the two parameters $\lambda_{\min}$ and $\lambda_{\max}$ act as safeguards required for the strong global convergence. The merit function f: $R^n \to R$ is defined as $\|F(x)\|_2^2/2$, and the parameter β is used to assure that the reduction of f is sufficient. Here, a linesearch technique [6] is employed to determine the step length, $\lambda^{(k)}$.

To define parallel Schwarz-type preconditioners we partition the finite element mesh $\mathcal{T}^h$. Let $\{\Omega_i^h, i = 1, ..., N\}$ be a non-overlapping subdomain partition whose union covers the entire mesh $\mathcal{T}^h$. We denote by $\mathcal{T}_i^h$ as the collection of mesh points in Ω_i^h. To obtain overlapping subdomains, we expand each subdomain Ω_i^h to a larger subdomain $\Omega_i^{h,\delta}$ with the boundary $\partial\Omega_i^{h,\delta}$. Here δ is an integer indicating the level of overlap. We assume that $\partial\Omega_i^{h,\delta}$ does not cut any elements of $\mathcal{T}^h$. Similarly, we denote by $\mathcal{T}_i^{h,\delta}$ as the collection of mesh points in $\Omega_i^{h,\delta}$.

Now, we define the subdomain velocity space as

$$V_i^h = \{\boldsymbol{v}^h \in V^h \cap (H^1(\Omega_i^{h,\delta}))^3 : \boldsymbol{v}^h = 0 \text{ on } \partial\Omega_i^{h,\delta}\}$$

and the subdomain pressure space as

$$P_i^h = \{p^h \in P^h \cap L^2(\Omega_i^{h,\delta}) : p^h = 0 \text{ on } \partial\Omega_i^{h,\delta} \backslash (\Gamma_{in} \cup \Gamma_{wall})\}.$$

On the physical boundaries, we impose Dirichlet conditions according to the original equations (1). On the artificial boundaries, we assume both $\boldsymbol{u} = 0$ and $p = 0$. Similar boundary conditions were used in [10, 12].

Let R_i: $V^h \times P^h \to V_i^h \times P_i^h$ be a restriction operator, which returns all degrees of freedom (both velocity and pressure) associated with the subspace $V_i^h \times P_i^h$. R_i is an $4n_i \times 4n$ matrix with values of either 0 or 1, where n and n_i are the total number of mesh points in $\mathcal{T}^h$ and $\mathcal{T}_i^{h,\delta}$, respectively, and $\sum_{i=1}^N 4n_i \geq 4n$. Note that for $P_1 - P_1$ elements, we have four variables per mesh point, three for the velocity and one for the pressure. Then, the extension operator R_i^T can be defined as the transpose of R_i. The multiplication of R_i (and R_i^T) with a vector does not involve any arithmetic operation, but does involve communication in a distributed memory parallel implementation. Using the restriction matrix, we write the one level additive Schwarz preconditioner in the matrix form (ASM) as

$$M_k^{-1} = \sum_{i=1}^{N} R_i^T J_i^{-1} R_i,$$

where J_i^{-1} is subspace inverse of $J_i = R_i J R_i^T$. We remark that the global-to-local restriction operator R_i collects the data from neighboring subdomains, and the local-to-global prolongation operator R_i^T sends partial solution to neighboring subdomains. In practice, to save the computational cost and the memory use, J_i^{-1} in M_k^{-1} are often replaced by an inexact solver, such as ILU with some levels of fill-ins.

4. Parallel software development

The heart of the blood flow simulator is the parallel fully implicit time-dependent flow solver based on the NKS algorithm as described in the previous section. The code is implemented by PETSc of the Argonne National laboratory [1], USA. Note that PETSc, a powerful tool for parallel computing, is designed for users to develop parallel scientific codes written in FORTRAN, C or C++ programming languages for solving large linear and nonlinear systems arising from PDEs in large-scale scientific applications. Several PETSc-based codes have been developed in a wide range

of applications in computational science and engineering, for example, nano-materials simulations, biomechanics calculations, fusion energy simulations, etc. The research represents the modern trend in scientific computing.

Once the geometric model of arteries is constructed using some image-to-surface tool, we employ a mesh generation software package, CUBIT of Sandia National laboratories, USA [34] to generate a 3D unstructured finite element mesh for the arteries and a mesh partitioning tool, ParMETIS of University of Minnesota, USA [13], to partition the mesh for the purpose of parallel processing. Later, we use a scientific visualization system, ParaView of Kitware and Sandia National Laboratories, USA [35] as a post-processor for data analysis. In addition, we develop some interface codes that convert the data files between two systems. One interface code transfers the CUBIT files for geometry, connectivity, and boundary conditions in the Abaqus format into a PETSc loadable file in the binary format. The other interface code combines CUBIT files for geometry and connectivity with the solution file exported from the fluid solver into a ParaView file in the VTK format.

5. Numerical results

In this section, we consider two test cases: (a) a long straight artery model [33] and (b) an end-to-side anastomosis model [17, 19] (See Figure 1), to check the correctness of our parallel fluid code as well as to evaluate its performance on a cluster of PCs. The detailed geometrical configurations and physiological flow conditions are introduced in the next subsection.

5.1. Test cases

5.1.1. A long straight artery model

We consider a long straight circular tube with a length of 3.0 cm and radius of 0.2 cm. As pulsatile flows, we impose a uniform periodic velocity $u(x,t) = \overline{V}(1 + \sin(2\pi t/T))$ with mean velocity $\overline{V}$ =13.5 cm/sec and period $T = 0.2$ sec for $x \in \Gamma_{in}$ as the inflow boundary condition, a stress-free condition as the outflow boundary condition, and a no-slip condition on the wall. In addition, we assume the flow is Poiseuille at the beginning of computation. For this model, we set a Reynolds number Re=135 based on the diameter of the tube and the mean velocity. In this case, a kinematic viscosity of 0.04 cm^2/sec is selected. We test oscillatory flows at both low and high frequencies, which correspond to the Womersley number $\alpha = 1$ and 10, respectively.

Figure 1: Case 1: a long straight artery model (top) and Case 2: an end-to-side anastomosis model (bottom).

Zamir [33] derives the analytical solution to this problem, known as the Womersley velocity profile for the axial velocity given by

$$u_1(r,t) = \frac{k_s}{4\mu}(r^2 - a^2) + \frac{ik_s a^2}{\mu\alpha^2}(1 - \frac{J_0(\zeta)}{J_0(\Lambda)})e^{i\omega t},$$

where k_s is the steady-state part of the pressure gradient and J_0 is the Bessel function of order zero of the first kind. Here, a is the diameter of the tube and the complex frequency parameter, $\Lambda = i^{3/2}\alpha$. For the situation of flows with low frequency rate, the solution can be further simplified using a series expansion of the Bessel function

$$u_1(r,t) \approx \frac{k_s}{4\mu}(r^2 - a^2)(1 + \cos\omega t). \tag{6}$$

Since the pressure gradient $k = k_s \cos(\omega t)$, compared to Eq. (6), we notice that at very low frequency, or in a limiting case of "zero frequency", the relation between flow and pressure becomes instantaneously the same as in the case of steady Poiseuille flow. But at high frequency the behavior of the velocity profile is different. For flows at high frequency rate, we first consider the flow near

the tube of the wall, i.e., $r/a \approx 1$, the analytical velocity can be written as

$$\begin{aligned} u_1(r,t) &= u_s(r) + u_\phi(r,t) = u_s(r) + u_{\phi R}(r,t) \\ &\approx \tfrac{k_s}{4\mu}(r^2 - a^2) + \tfrac{2\sqrt{2}k_s}{4\mu\alpha}(ar - a^2)(\cos\omega t + \sin\omega t) \\ &\approx \tfrac{k_s}{4\mu}[(r^2 - a^2) + \tfrac{2\sqrt{2}}{\alpha}(ar - a^2))(\cos\omega t + \sin\omega t)]. \end{aligned}$$

On the other hand, for the flow around the center of the tube, i.e., $r/a \approx 0$

$$\frac{u_\phi(r,t)}{\hat{u_s}} = \frac{-4}{\Lambda^2} e^{i\omega t}.$$

The real and imaginary parts are given by

$$\begin{aligned} \frac{u_{\phi R(r,t)}}{\hat{u_s}} &\approx \tfrac{4}{\alpha^2}\sin\omega t, \\ \frac{u_{\phi I(r,t)}}{\hat{u_s}} &\approx \tfrac{-4}{\alpha^2}\cos\omega t, \end{aligned}$$

where $\hat{u_s} = -\frac{k_s}{4\mu}a^2$ is the corresponding maximal velocity in Poiseuille flow. Thus, the analytical velocity

$$\begin{aligned} u(r,t) &= u_s(r) + u_\phi(r,t) = u_s(r) + u_{\phi R}(r,t) \\ &\approx \tfrac{k_s}{4\mu}(r^2 - a^2) + \tfrac{k_s a^2}{\alpha^2\mu}\sin\omega t. \end{aligned} \tag{7}$$

From the above results at high frequency the flow near the wall of the tube is affected differently from the flow near the center of the tube, thus the parabolic characteristic of the velocity profile is distorted and the 90 degree phase difference between velocity and pressure is expected.

5.1.2. An end-to-side anastomosis model

For the end-to-side anastomosis model as shown in Figure 2, the diameter of the host artery is 1.75 mm while the diameter of the graft artery is 2.8 mm. The graft branching angle is 5 degrees and the ratio of the length of the hood to the diameter of the host artery is 5 to 1.

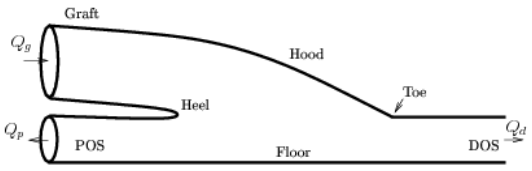

Figure 2: Specific regions of the end-to-side graft model.

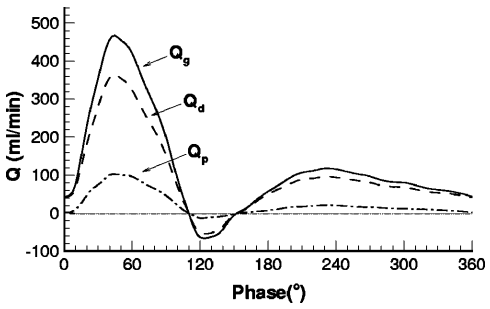

Figure 3: Physiological flow waveforms at inlet (Q_g) and outlets (Q_p , Q_d) [17].

For the case of Poiseuille flow, in vivo values, the graft input flow rate = 1.9 ml/sec, the dynamic viscosity = 0.035 g/s/cm, and the density = 1.05 g/ml. The division ratio of flow between the proximal outlet segment (POS) and the distal outlet segment (DOS) is set at 20:80 for a Reynolds number of 208 based on the host artery diameter. On the other hand, for the case of Pulsatile flows, Dirichlet type conditions at the inflow and the POS are set according to the physiological flow waveforms as shown in Figure 3. The DOS is set to be the stress-free boundary condition. The frequency of the pulsatile waveform $\omega = 1.4$Hz and the corresponding Womersley number is 2.8. The mean flow rate ratio is about 20:80 (The typical flow rate ratio during the systolic phase is 22:78). The Reynolds number, which is based on the host artery diameter, has a maximum value of 850 and a mean value of 222. We take the mean input velocity as the characteristic velocity, so the total period, T, is 43.2 and the kinematic viscosity is the inverse of the mean Reynolds number.

5.2. Parallel fluid code validation

Before validating our parallel fluid code, we first perform the mesh resolution independent test. We compute directly the steady-state solution to the NS equation. For the long straight artery model, we test a sequence of four meshes with different sizes (the total number of elements ranges from about 55,000 to 202,000). As a result, we find the numerical velocity profiles obtained by using a mesh with 103,625 elements and 19,930 nodes is almost distinguishable with the analytical solution. Similarly for the end-to-side graft model, five meshes are tested ranging from 38,000 to 788,000 elements. We also observe that the numerical velocity profiles obtained by using a mesh with 103,625 elements and 19,930 nodes at different observation cross sections are consistent with the experimental data. Hence, these two meshes are used for the rest of the numerical experiments.

For the long straight artery model, we verify whether the incompressibility condition is satisfied by first computing numerically the outflow flow rate and then comparing it with the inflow flow rate with respect to time during the first period. As shown in Figure 4, they are quite consistent.

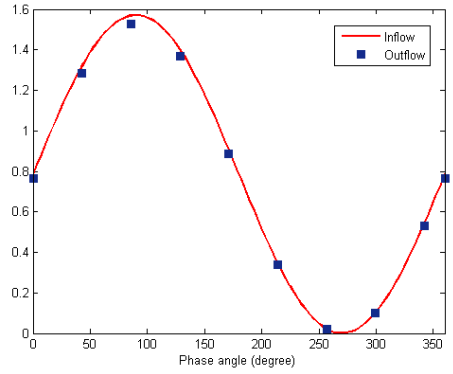

Figure 4: A comparison of the analytical flow rate at the inlet boundary and the computed flow rate at the outlet boundary at low frequency.

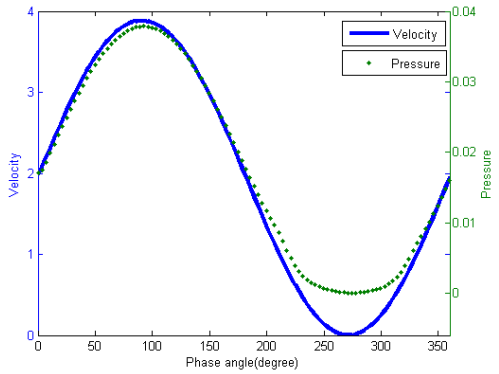

Figure 5: Low frequency case: Variation of velocity, compared with variation of pressure at the center of the tube with in an oscillatory cycle.

As mentioned in Subsection 5.1.1, for the low frequency flows, the peak of the axial velocity (Eq. (6)) is in phase with the peak of pressure gradient. But the peak of the axial velocity (Eq. (7)) lags the peak of the pressure gradient by 90 degrees at high frequency. As observed in Figures 5 and 6, neither the phase nor the amplitude of these peaks profiles correspond with the peaks of the pressure clearly due to the inertia of the fluid. In addition, the difference of the phase angle at

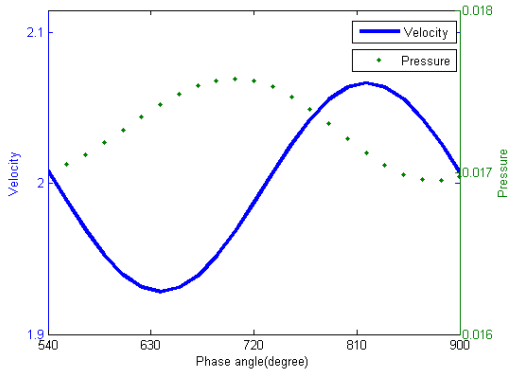

Figure 6: High frequency case: Variation of velocity, compared with variation of pressure at the center of the tube with in an oscillatory cycle.

low frequency to high frequency is near 0 to 90 degrees.

For the end-to-side graft model, we compare the numerical velocity profiles with experimental data [19] at the cross section near the proximal end of the anastomotic region in the x-direction and the y-direction in the symmetry plane in the phases $t = T/8$, $T/4$, $T/3$, and $2T/3$ in Figures 7 and 8, respectively. Note that these four phases represent the peak systole, the late systole, the early diastole, and the mid-diastole in order. It is observed that the computational results do agree qualitatively with the experimental data, especially that we can catch the location of the reverse flow zone.

Figure 9 displays the computed streamlines at these four phases. We observe that the flows decelerate during the late systole phase. Some vortices form near the hood and the direction of flows is reversed at the early diastole phase and the streamlines are more chaotic at this phase than these at other phases. The flow patterns at the mid-diastole and the peak systole are similar but the flow speed is slower at the mid-diastole phase. Figure 10 depicts the computed pressure distribution at these four phases.

5.3. Parallel performance study

In our numerical simulation, at each time step, we employ NKS to solve (3) with the previous time step solution as the initial guess. We claim the intermediate solution converges when the stopping condition for Newton

$$\|F(x^{(k)})\| \leq \max\{10^{-6}\|F(x^{(0)})\|, 10^{-10}\}$$

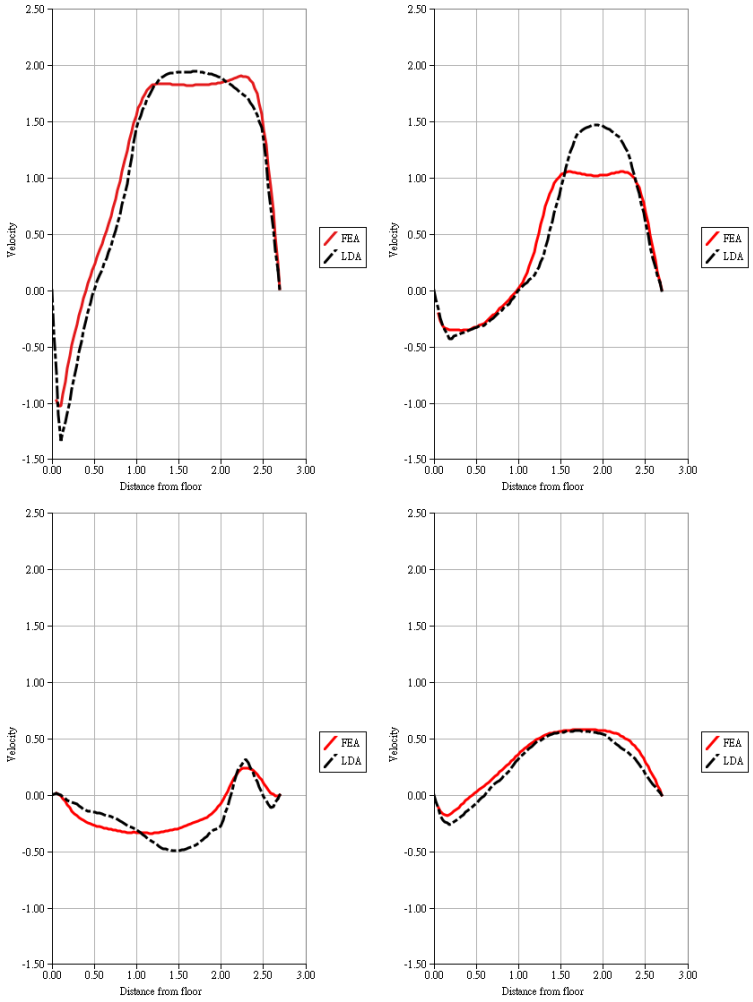

Figure 7: A comparison of the computed velocity profile (FEA) in x-direction and experimental data (LDA) at $t = T/8$ (top left), $T/4$ (top right), $T/3$ (bottom left), and $2T/3$ (bottom right) at a cross section near the proximal end of anastomotic region.

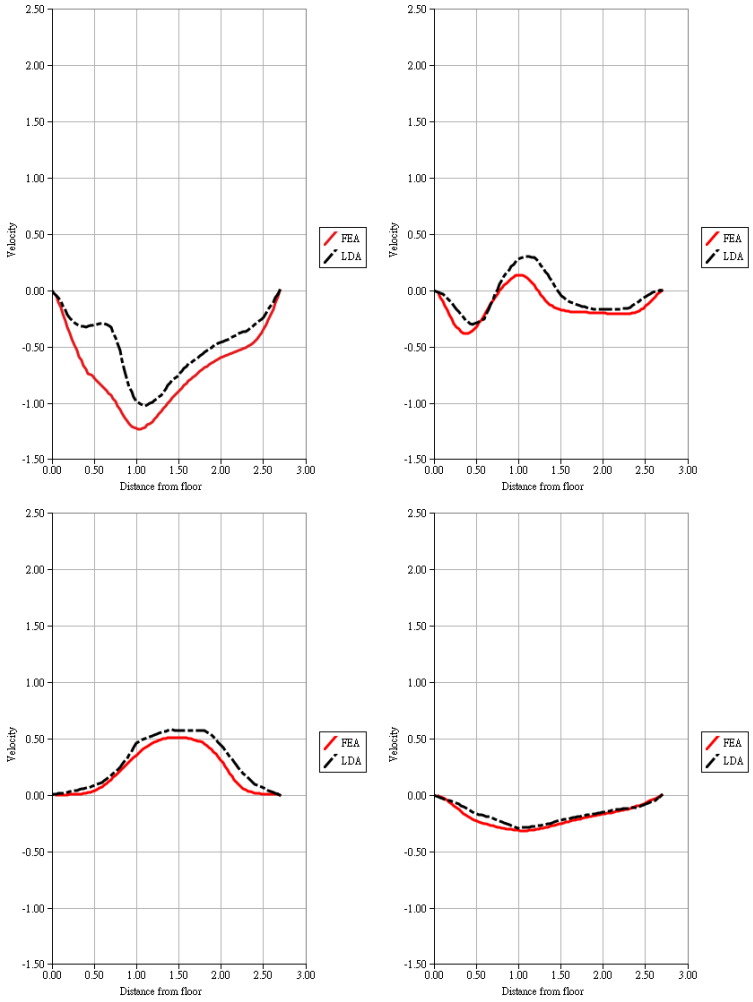

Figure 8: A comparison of the computed velocity profile (FEA) in y-direction and experimental data (LDA) at $t = T/8$ (top left), $T/4$ (top right), $T/3$ (bottom left), and $2T/3$ (bottom right) at a cross section near the proximal end of anastomotic region.

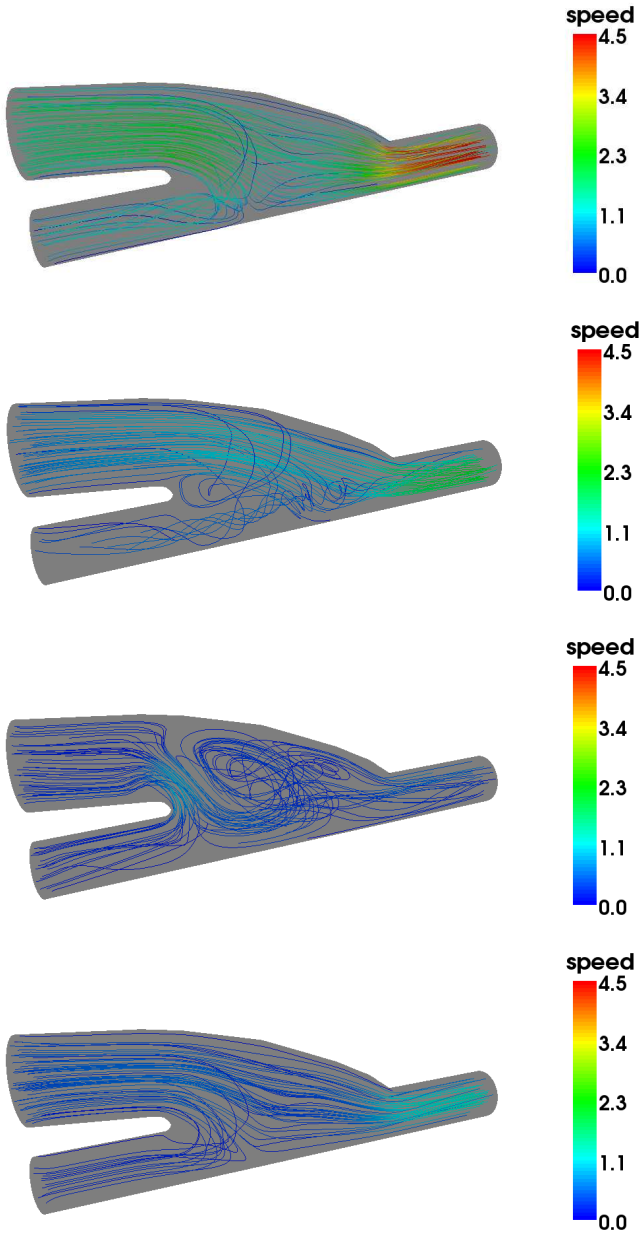

Figure 9: The computed streamlines at $t = T/8$, $T/4$, $T/3$, and $2T/3$ (from top to bottom).

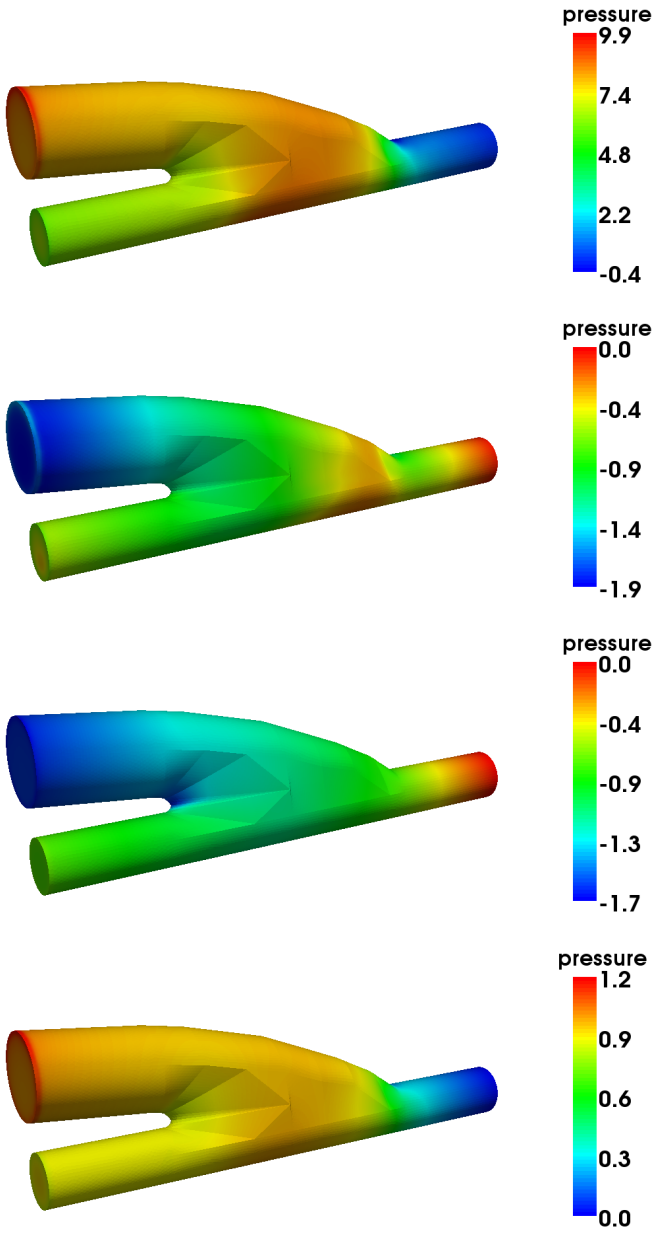

Figure 10: The computed pressure distribution at $t = T/8$, $T/4$, $T/3$, and $2T/3$ (from top to bottom).

is satisfied. A cubic linesearch technique [6] is employed to determine the step length $\lambda^{(k)}$, with $\beta = 10^{-4}$, $\lambda_{\min} = 1/10$ and $\lambda_{\max} = 1/2$. A right additive Schwarz preconditioned GMRES with a zero initial guess is employed to solve the Jacobian system. The components of the Jacobian corresponding to Galerkin finite element formulations are analytically constructed and other components associated with stabilized terms are approximately computed by using multicolored finite differences. The accuracy of the solution to the Jacobian systems is controlled by the parameter, η_k, in the following condition,

$$\|F(x^{(k)}) + (J_k(x^{(k)})M_k^{-1})(M_k s^{(k)})\| \leq \max\{\eta_k \|F(x^{(k)})\|, 10^{-10}\}.$$

Here $\eta_k = 10^{-4}$. And the maximum number of GMRES iterations is limited to be at 200.

Figure 11 shows the history of the number of Newton iterations at each time step, and Figure 12 shows the average number of GMRES iterations at each time step. The end-to-side anastomosis model is considered and the results are obtained by using 48 processors. Since we use the solution of the previous time step as the initial guess, Newton is quite robust, i.e., only requires at most three iterations to reach convergence. On the other hand, from Figure 12, except for the beginning of the simulation, we notice that the Jacobian system is more ill-conditioned during the reverse flow phase, as a result NKS requires more linear iterations in this phase than in other phases.

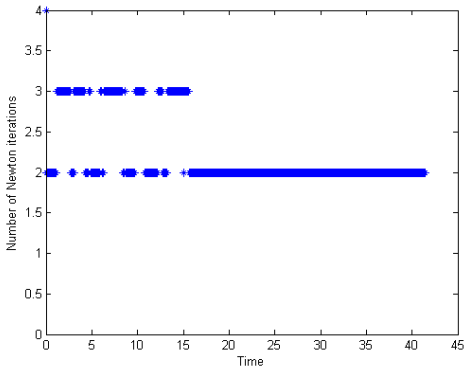

Figure 11: Number of Newton iterations at each time step.

Next, some parameter studies and the parallel performance of the software are discussed. We report the results based on 10 time steps performed with a fixed $\Delta t = 0.1$ for several different number of processors. The execution time reported in seconds only takes into account for the

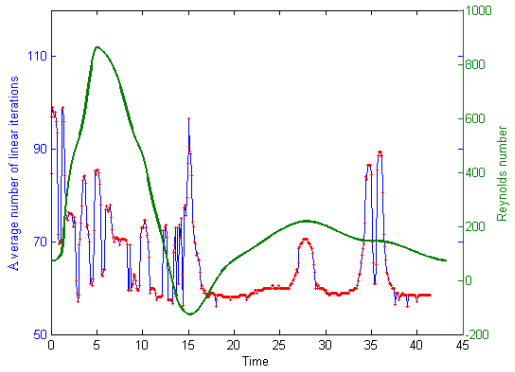

Figure 12: Average number of linear iterations and the value of Reynolds number at each time step.

computing time spent on the nonlinear solve, which includes the function evaluations, the Jacobian constructions, and the inner Jacobian solves.

We investigate how the parameters involved, such as the quality of subdomain solve and and the overlapping size for ASM, affect the overall performance of the NKS algorithm. We test the cases using the LU decomposition and the incomplete LU (ILU) decomposition with different levels of fill-ins, $k = 0, 1, 2, 3$ and vary the overlapping size $\delta = 1, 2, 3, 4$.

We observe that by increasing the size of ASM overlapping or the levels of fill-ins of ILU, the average number of GMRES iterations is decreased. However, typically, smaller overlapping size and fewer levels of ILU fill-ins produces better timing results. Meanwhile, although when LU is used as the subdomain solver, the number of iterations is smaller than when ILU is used as the subdomain solver, the overall timing results, NKS with ILU is much faster than that of NKS with LU. The best combinations of the level of ILU fill-ins and the overlapping size in terms of the execution time with respect to the number of processors are summarized in Table 1.

Finally, to evaluate the parallel performance of our blood flow simulation, we consider the parallel efficiency defined as

$$E_f = \left(\frac{16}{N_{np}}\right)\frac{T_{16}}{T_{np}}.$$

From Table 1, the parallel efficiency of our parallel fluid solver achieves at least 50% with up to 64 processors and it is degraded when 128 processors is used. Multilevel methods will be needed when the number of processors is larger [3].

np	ILU(k), δ	NNI, ANLI	time	E_f (%)
16	1, 1	2.5, 159.7	995.9	100
32	1, 1	2.6, 170.7	690.3	72.10
64	1, 2	2.5, 167.5	473.8	52.60
128	2, 1	2.4, 169.0	432.7	28.50

Table 1: Parallel efficiency. k: the level of ILU fill-ins, δ: the overlapping size for ASM, NNI: the number of nonlinear iterations per time step, and ANLI: the average number of linear iterations per Newton at each time step.

6. Concluding remarks

In this work, we developed a computer simulation system for blood flows in large arteries. The algorithms are based on the fully coupled and fully implicit scalable Newton-Krylov-Schwarz framework, and the software are developed on top of PETSc and several other recently introduced open source packages. We validated the numerical algorithms and the software by considering two representative model problems; namely a straight artery case and an end-to-side case. Our studies showed good qualitative agreements between numerical solutions, analytical solutions, as well as experimental data. Moreover, our solver achieved above 50% of parallel efficiency with up to 64 processors on a cluster of PCs.

References

[1] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang, Portable, Extensible Toolkit for Scientific Computation (PETSc) home page, http://www.mcs.anl.gov/petsc, 2009.

[2] M. Behr, D. Arora, O. Coronado, M. Pasquali, Models and finite element techniques for blood flow simulation, Int. J. Comput. Fluid Dyn. 20 (2006) 175-181.

[3] A. Barker, X.-C. Cai, Two-level Newton and hybrid Schwarz preconditioner for fluid-structure interaction, (2009) submitted.

[4] A. Barker, X.-C. Cai, Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling, J. Comput. Phys. 229 (2010), 642–659.

[5] X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, D.P. Young, Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation, SIAM J. Sci. Comput. 19 (1998), 246-265.

[6] J. Dennis, R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM, Philadelphia, 1996.

[7] F.C. Fung, Biomechanics: Circulation, Springer-Verlag, New York, 1997.

[8] L.P. Franca, S.L. Frey, Stabilized finite element method: II. The incompressible Navier-Stokes equation, Comput. Methods Appl. Mech. Engrg. 99 (1992) 209–233.

[9] M.D. Gunzburger, Finite Element Methods for Viscous Incompressible Flows, Academics Press, New York, 1989.

[10] F.-N. Hwang, X.-C. Cai, Parallel fully coupled Schwarz preconditioners for saddle point problems, Electron. Trans. Numer. Anal. 22 (2006) 146-162.

[11] C. Johnson, Numerical Solution of Partial Differential Equations by the Finite Element Method, Cambridge University Press, Cambridge, 1987.

[12] A. Klawonn, L.F. Pavarino, Overlapping Schwarz methods for mixed linear elasticity and Stokes problems, Comput. Methods Appl. Mech. Engrg. 165 (1998) 233–245.

[13] G. Karypis, R. Aggarwal, K. Schloegel, V. Kumar, S. Shekhar, METIS home page, http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

[14] Y. Khunatorn, S. Mahalingam, C. DeGroff, R. Shandas, A fluid dynamic study of the total cavopulmonary connection in the Fontan operation, Proceedings of the 11th International Conference on Mechanics in Medicine and Biology, April, 2000.

[15] Y. Khunatorn, S. Mahalingam, C. G. DeGroff, R. Shandas, Influence of connection geometry and SVC-IVC flow rate ratio on flow structures within the total cavopulmonary connection: A numerical study, J. Biomech. Eng. 124 (2002) 364–377.

[16] D.A. Knoll, D.E. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comp. Phys. 193 (2004) 357–397.

[17] M. Lei, D.P. Giddens, S.A. Jones, F. Loth, H. Bassiouny, Pulsatile flow in an end-to-side vascular graft model: comparison of computations with experimental data, J. Biomech. Eng. 123 (2001) 80–87.

[18] R. Löhner, J. Cebral, O. Soto, P. Yim, J.E. Burgess, Applications of patient-specific CFD in medicine and life sciences, Int. J. Numer. Meth. Fluids 43 (2003) 637–650.

[19] F. Loth, S.A. Jones, D.P. Giddens, H.S. Bassiouny, S. Glagov, C.K. Zarins, Measurements of velocity and wall stress inside a PTFE vascular graft model under steady flow conditions, J. Biomech. Eng. 119 (1997) 187–194.

[20] J. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.

[21] A. Quarteroni, M. Tuveri, A. Veneziani, Computational vascular fluid dynamics: problems, models, and methods, Comp. and Visual. Sci. 2 (2000) 163–197.

[22] A. Quarteroni, A. Valli, Domain Decomposition Methods for Partial Differential Equations, Oxford University Press, Ofxord, 1999.

[23] J.T. Ottesen, M.S. Olufsen, J.K. Larsen, Applied Mathematical Models in Human Physiology, SIAM, Philadelphia, 2004.

[24] M. Oshima, R. Torii, T. Kobayashi, N. Taniguchi, K. Takagi, Finite element simulation of blood flow in the cerebral artery, Comput. Methods Appl. Mech. Engrg. 191 (2001) 661–671.

[25] J.N. Reddy, D.K. Gartling, The Finite Element Method in Heat Transfer and Fluid Dynamics, CRC Press, Florida, 2000.

[26] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsysmetric linear system, SIAM J. Sci. Stat. Comp. 7 (1986) 856–869.

[27] B. Smith, P. Bjørstad, and W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 1996.

[28] C.A. Taylor, T.J.R Hughes, C.Z Zarins, Finite element modeling of blood flow in arteries, Comput. Meth. Appl. Mech. Engrg. 158 (1998) 155–196.

[29] A. Toselli, O. Widlund, Domain Decomposition Methods – Algorithms and Theory, Springer-Verlag, Berlin, 2005.

[30] I.E. Vignon-Clementel, C.A. Figueroa, K.E. Jansen, C.A. Taylor, Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries, Comput. Methods Appl. Mech. Engrg. 195 (2006) 3775–3776.

[31] K. Yokoi, F. Xiao, H. Liu, K. Fukasaku, Three-dimensional numerical simulation of flows with complex geometries in a regular Cartesian grid and its applications to blood flow in cerebral artery with multiple aneurysms, J. Comput. Phys. 202 (2005) 1–19.

[32] X. Yue, F.-N. Hwang, R. Shandas, X.-C. Cai, Simulation of branching blood flows on parallel computers, Biomed. Sci. Instrum. 40 (2004) 325-330.

[33] M. Zamir, The Physics of Pulsatile Flow. Springer-Verlag, Berlin, 2000.

[34] Online CUBIT User's Manual, http://cubit.sandia.gov/documentation.html, Sandia National Laboratories, 2009.

[35] ParaView homepage. http://www.paraview.org.