


A parallel non-nested two-level domain decomposition method for simulating blood flows in cerebral artery of stroke patient

Rongliang Chen^{1,2} | Bokai Wu¹ | Zaiheng Cheng¹ | Wen-Shin Shiu¹ | Jia Liu¹ | Liping Liu³ | Yongjun Wang³ | Xinhong Wang⁴ | Xiao-Chuan Cai⁵ 

¹Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

²Shenzhen Key Laboratory for Exascale Engineering and Scientific Computing, Shenzhen, China

³Department of Neurology, Beijing Tiantan Hospital, Capital Medical University, Beijing, China

⁴The Second Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou, China

⁵Department of Mathematics, University of Macau, Macau, China

Correspondence

Xiao-Chuan Cai, Department of Mathematics, University of Macau, Macau, China.
Email: xccai@um.edu.mo

Funding information

National Key R&D Program of China, Grant/Award Number: 2016YFB0200601; Shenzhen grant, Grant/Award Numbers: JCYJ20170818153840322, ZDSYS201703031711426

Abstract

Numerical simulation of blood flows in patient-specific arteries can be useful for the understanding of vascular diseases, as well as for surgery planning. In this paper, we simulate blood flows in the full cerebral artery of stroke patients. To accurately resolve the flow in this rather complex geometry with stenosis is challenging and it is also important to obtain the results in a short amount of computing time so that the simulation can be used in pre- and/or post-surgery planning. For this purpose, we introduce a highly scalable, parallel non-nested two-level domain decomposition method for the three-dimensional unsteady incompressible Navier-Stokes equations with an impedance outlet boundary condition. The problem is discretized with a stabilized finite element method on unstructured meshes in space and a fully implicit method in time, and the large nonlinear systems are solved by a preconditioned parallel Newton-Krylov method with a two-level Schwarz method. The key component of the method is a non-nested coarse problem solved using a subset of processor cores and its solution is interpolated to the fine space using radial basis functions. To validate and verify the proposed algorithm and its highly parallel implementation, we consider a case with available clinical data and show that the computed result matches with the measured data. Further numerical experiments indicate that the proposed method works well for realistic geometry and parameters of a full size cerebral artery of an adult stroke patient on a supercomputers with thousands of processor cores.

KEYWORDS

domain decomposition method, finite element on unstructured meshes, full cerebral artery with stenosis, fully implicit method, non-nested coarse space, parallel processing

1 | INTRODUCTION

Hemodynamic analysis of patient-specific artery is useful for the understanding and diagnosis of various vascular diseases, such as angiosclerosis and angiostenosis, as well as for surgery planning. Several statistical studies show that the hemodynamic flow patterns and parameters, such as wall shear stress (WSS), flow separation, regurgitation, vortex, etc.,

are considered to be closely related to the formation and progression of the cerebrovascular diseases.^{1,2,3} Four-dimensional magnetic resonance imaging (4D MRI) and computational fluid dynamic (CFD) are the two major methods to study the hemodynamic. The spatial and temporal resolutions of 4D MRI are 2 to 3 mm and 40 to 50 ms, respectively. This level of resolution is unable to capture certain small scale features of the fluid and the scan time of the 4D MRI is relatively long.^{4,5} On the other hand, CFD is able to provide higher resolution than 4D MRI, except that it often takes more time to obtain CFD results due to the high computational cost.⁶

CFD based blood flow simulation is increasingly used to obtain the hemodynamic properties of individual patient. For example, Dedè et al studied the blood flow in an idealized left human heart.⁷ Buoso et al introduced a parameterized reduced-order method for the noninvasive functional evaluation of coronary artery diseases.⁸ Stoter et al used the Navier-Stokes/Darcy equations to simulate the blood flow of a patient-specific human liver.⁹ Some recent reviews of patient-specific blood flow simulations can be found in References 10 and 11. For the cerebral blood flow, there are some recent works for patients with aneurysms^{12,13} and stenosis.^{14,15,16} Due to the computational cost and convergence issues, most simulations for the cerebral artery consider only a small segment or a few branches of the artery, or using simplified geometry such as a bifurcating tube to represent the artery. For example, in Reference 17 a short, patient-specific, segment of the cerebral artery with a zero-one-dimensional boundary condition was considered for the hemodynamic study. In Reference 18, a Navier-Stokes model for the left internal carotid artery coupled with a one-dimensional Euler model for the circle of Willis is used to simulate the blood flow in the cerebral artery. Some very large scale simulations were carried out in References 19 and 20, in which the entire cerebral arteries with hundreds of branches were included under the assumption that the arteries are cylindrical. In this paper, we push the limit further to include the entire cerebral artery with patient-specific geometry and parameters.

Parallel processing is unavoidable for such large scale simulations, there are several papers devoted to the development of parallel algorithms for blood flow simulations, for example, Forti et al introduced a parallel block preconditioner for a two-branch artery case.²¹ In paper,²² a parallel one-level Newton-Krylov-Schwarz approach was introduced for an idealized artery with the stress-free outlet boundary condition and the work was extended later to multilevels in References 23 and 24. Randles et al introduced a parallel lattice Boltzmann method for the hemodynamics simulation of the systemic arterial tree that scales to 1 572 864 processor cores on a Blue Gene/Q supercomputer.^{25,26}

In this paper, we study a highly scalable algorithm with which the simulation time can be considerably reduced. We introduce a scalable parallel algorithm based on a two-level domain decomposition method for the simulation of patient-specific cerebral blood flows with the impedance boundary condition on the outlet boundaries. Two-level overlapping Schwarz methods have been used to solve many problems, such as PDE constrained optimization problems,^{27,28} fluid-structure interaction problems,^{29,23} shallow water equations,³⁰ pyramidal quantum dot simulations,³¹ etc. In Reference 23, the authors introduced a two-level method for the blood flow simulation where the stress-free boundary condition is used as the outlet boundary condition. In most two-level methods, the fine mesh and the coarse mesh are nested, and the coarse mesh is obtained by the coarsening of the fine mesh, and for such a situation the interpolation and restriction matrices come naturally, without much additional cost, from the coarsening algorithm. In such an approach, the size of the coarse mesh and the partition of the coarse mesh are both difficult to control, especially when the geometry is complex and the number of processor cores is large. In our two-level method, we do not require that the fine mesh and coarse mesh are nested. They are generated independently. The coarse mesh is partitioned independent of the fine mesh, and the problem is solved using a subset of processor cores used for the fine mesh problem. The coarse solution is interpolated to the fine space using radial basis functions which are able to handle the situation when some fine mesh nodes are outside of the coarse mesh domain.

The other challenging issue to be considered in the paper is the use of the impedance outflow boundary condition. The condition is physiologically more accurate, but it is time-dependent, and non-local since it is an integral taken on the outlet surface. The dense block matrix could destroy the overall performance if not considered carefully. Most blood flow simulations only consider a small segment of the blood vessel so that the overall cost of the computation cost is low, however in such a situation the outlet surface is often large and the matrix from the impedance outflow condition is quite large and dense. This issue makes the impedance condition not a popular choice. In this paper, since we consider the full size cerebral artery, all the outlets are very small (0.8 ~ 1.5 mm in diameter). As a result, the dense matrices from the impedance condition are all reasonably small, therefore not causing major problems in load balancing, large condition numbers, and difficulty in mesh partitioning.

The rest of the paper is organized as follows. In section 2, we describe the system of incompressible Navier-Stokes equations with impedance boundary conditions as a model for the blood flow, and the corresponding finite element discretization is also discussed in detail. A two-level domain decomposition solver with a non-nested coarse space and its

parallel implementation are introduced in section 3. Some numerical experiments are presented in section 4 to show the efficiency of the proposed method and some final remarks are given in section 5.

2 | MATHEMATICAL MODEL AND DISCRITIZATION

We consider the Newtonian model of the blood flow governed by the time-dependent incompressible Navier-Stokes equations

$$\begin{cases} \rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) = -\nabla p + \mu \Delta \mathbf{u}, & \text{in } \Omega, \\ \nabla \cdot \mathbf{u} = 0, & \text{in } \Omega. \end{cases} \quad (1)$$

Here $\Omega \in R^3$ is the artery as shown in Figure 1, $\mathbf{u} = (u, v, w)^T$ is the velocity, ρ is the blood density, p is the pressure, and μ is the viscosity coefficient. To make the system (1) solvable, we need to impose boundary conditions on the inlets Γ_{inlet} , outlets Γ_{outlet} , and the wall Γ_{wall} ; see Figure 1.

The choice of boundary conditions is important in cerebral blood flow simulations, as the fluid behavior is influenced greatly by the conditions upstream and downstream. Numerous studies have demonstrated that there are drastic differences in the computed flow field with different boundary conditions,^{32,33,34,35,36,37} especially the outflow boundary conditions. Briefly speaking, there are four popular outflow boundary conditions, namely: (a) stress-free boundary condition; (b) constant pressure boundary condition; (c) resistance boundary condition, and (d) impedance boundary condition. The resistance and impedance boundary conditions are derived by modeling the peripheral arteries with resistance and compliance, and they require certain knowledge of the peripheral arterial network characteristics. In this paper, we focus on the impedance boundary condition.

On the artery wall, we apply a no-slip boundary condition; that is,

$$\mathbf{u} = \mathbf{0}, \quad \text{on } \Gamma_{\text{wall}}$$

and a patient-specific time-dependent flow rate $Q_{\text{inlet}}(t)$ is imposed on the inlets. The total flow rate is distributed to the four inlets based on their relative areas, that is the flow rate for each inlet $Q_{\text{inlet}}^i(t) = (A_{\text{inlet}}^i/A_{\text{inlet}})Q_{\text{inlet}}(t)$ ($i = 1, \dots, 4$), where A_{inlet} and A_{inlet}^i are the total area of the four inlets and the area of the i th inlet, respectively. For each of the outlet boundary, a three-element Windkessel model³⁸ is used which can be represented by the following ordinary differential equation

$$C \frac{dp(t)}{dt} + \frac{1}{R_d} p(t) = RC \frac{dQ_{\text{outlet}}(t)}{dt} + Q_{\text{outlet}}(t) + \frac{R}{R_d} Q_{\text{outlet}}(t) + \frac{p_b}{R_d}, \quad (2)$$

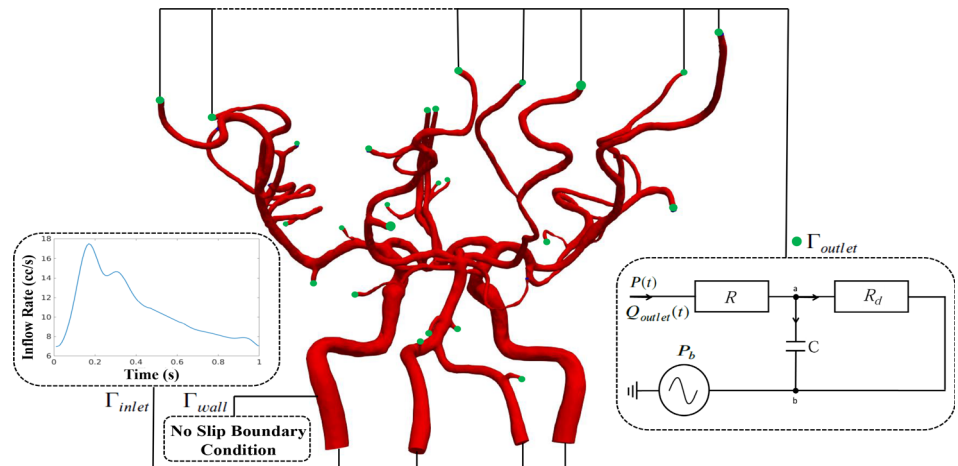


FIGURE 1 A cerebral artery with multiple inlets and outlets. A flow rate is used as the inlet condition, and an impedance boundary condition is applied at the outlets

where R , R_d , C are the resistances and capacitance shown in Figure 1. $p(t)$ and $Q_{\text{outlet}}(t)$ are the pressure and flow rate at the outlets, respectively. p_b is the pressure at the downstream. The analytic solution of (2) is

$$\begin{aligned} p(t) &= [p(0) - RQ_{\text{outlet}}(0) - p_b]e^{-\frac{t}{\tau}} + p_b + RQ_{\text{outlet}}(t) \\ &\quad + \int_0^t \frac{e^{-(t-s)/\tau}}{C} Q_{\text{outlet}}(s) ds \\ &= [p(0) - RQ_{\text{outlet}}(0) - p_b]e^{-\frac{t}{\tau}} + p_b + R \int_{\Gamma_{\text{outlet}}} \mathbf{u}(t) \cdot \mathbf{n} d\Gamma \\ &\quad + \int_0^t \frac{e^{-(t-s)/\tau}}{C} \left(\int_{\Gamma_{\text{outlet}}} \mathbf{u}(s) \cdot \mathbf{n} d\Gamma \right) ds, \end{aligned} \quad (3)$$

where $\tau = R_d C$, $p(0)$ and $Q_{\text{outlet}}(0)$ are the initial pressure and flow rate, respectively. \mathbf{n} is the outward unit surface normal to Γ_{outlet} . When there are multiple outlets, we split the resistances and capacitances according to the rules for a parallel circuit and the area of the corresponding outlet, that is

$$R_i = \frac{(A_{\text{outlet}})^{1.5}}{(A_{\text{outlet}}^i)^{1.5}} R_{\text{total}}, \quad C_i = \frac{A_{\text{outlet}}^i}{A_{\text{outlet}}} C_{\text{total}},$$

where A_{outlet} is the sum of all the areas of the outlets, A_{outlet}^i is the area of the i th outlet, and R_{total} and C_{total} are the total resistance and capacitance, respectively.³⁹

The Navier-Stokes Equations (1) is discretized by a stabilized $P_1 - P_1$ finite element method⁴⁰ in space. To describe the finite element method, we first define the trial and weighting function spaces as

$$\begin{aligned} \mathcal{U} &= \left\{ \mathbf{u}(\cdot, t) \mid \mathbf{u}(\cdot, t) \in [H^1(\Omega)]^3, \mathbf{u}(\cdot, t) = \mathbf{g} \text{ on } \Gamma_{\text{inlet}} \right\}, \\ \mathcal{U}_0 &= \left\{ \mathbf{u}(\cdot, t) \mid \mathbf{u}(\cdot, t) \in [H^1(\Omega)]^3, \mathbf{u}(\cdot, t) = \mathbf{0} \text{ on } \partial\Omega \right\}, \\ \mathcal{P} &= \left\{ p(\cdot, t) \mid p(\cdot, t) \in L^2(\Omega) \right\}, \end{aligned}$$

where \mathbf{g} is the Dirichlet boundary condition applied on the inlet Γ_{inlet} . Then, the weak form takes the form: Find $\mathbf{u} \in \mathcal{U}$, $p \in \mathcal{P}$ such that

$$\left\{ \begin{aligned} \rho \left(\int_{\Omega} \frac{\partial \mathbf{u}}{\partial t} \Phi d\Omega + \int_{\Omega} (\mathbf{u} \cdot \nabla) \mathbf{u} \cdot \Phi d\Omega \right) &= \int_{\Omega} p \nabla \cdot \Phi d\Omega - \mu \int_{\Omega} \nabla \mathbf{u} \cdot \nabla \Phi d\Omega \\ &\quad + \int_{\Gamma_o} (-p \mathbf{I} + \mu \nabla \mathbf{u}) \cdot \Phi \cdot \mathbf{n} d\Gamma, \\ \int_{\Omega} (\nabla \cdot \mathbf{u}) \varphi d\Omega &= 0 \end{aligned} \right. \quad (4)$$

holds for all $\Phi \in \mathcal{U}_0$ and $\varphi \in \mathcal{P}$, where \mathbf{I} is a 3×3 identity matrix. To apply the impedance boundary condition, we replace p in the last term of (4) by Equation (3).

We cover the computational domain with an unstructured tetrahedral mesh $\mathcal{T}^h = \{K\}$. Denote the mesh on the outlet boundary as $\Gamma_{\text{outlet}}^h = \{\Gamma_{\text{outlet}}^K\}$. The finite dimensional trial and weighting spaces can then be established as

$$\begin{aligned}\mathcal{U}^h &= \left\{ \mathbf{u}^h(\cdot, t) \mid \mathbf{u}^h(\cdot, t) = \sum_{i=1}^{N_u} \Phi_i^h \mathbf{u}_i^h(\cdot, t), \mathbf{u}^h(\cdot, t) = \mathbf{g} \text{ on } \Gamma_{\text{inlet}} \right\}, \\ \mathcal{U}_0^h &= \left\{ \mathbf{u}^h(\cdot, t) \mid \mathbf{u}^h(\cdot, t) = \sum_{i=1}^{N_u} \Phi_i^h \mathbf{u}_i^h(\cdot, t), \mathbf{u}^h(\cdot, t) = \mathbf{0} \text{ on } \partial\Omega \right\}, \\ \mathcal{P}^h &= \left\{ p^h(\cdot, t) \mid p^h(\cdot, t) = \sum_{i=1}^{N_p} \varphi_i^h p_i^h(\cdot, t) \right\},\end{aligned}$$

where $\mathbf{u}_i^h \in R^3$, $p_i^h \in R$ are the nodal values of the velocity and pressure functions. N_u and N_p are the number of nodes for the velocity and the pressure, respectively. Each of the three components of Φ_i^h and φ_i^h are the basis functions which are piecewise linear continuous functions. Since the $P_1 - P_1$ element does not satisfy the Ladyzenskaja-Babuska-Brezzi (LBB) condition, we need to add suitable stabilization terms. For this purpose, we employ the stabilization technique introduced in Reference 40. The semi-discrete stabilized finite element formulation of (4) is given as follows: Find $\mathbf{u}^h \in \mathcal{U}^h$, $p \in \mathcal{P}^h$ such that

$$\left\{ \begin{aligned} & \rho \left(\int_{\Omega} \frac{\partial \mathbf{u}^h}{\partial t} \cdot \Phi^h d\Omega + \int_{\Omega} (\mathbf{u}^h \cdot \nabla) \mathbf{u}^h \cdot \Phi^h d\Omega \right) = \\ & \quad \int_{\Omega} p^h \nabla \cdot \Phi^h d\Omega - \mu \int_{\Omega} \nabla \mathbf{u}^h \cdot \nabla \Phi^h d\Omega \\ & \quad + \int_{\Gamma_{\text{outlet}}^h} \left(- \left([p(0) - RQ(0) - p_b] e^{-\frac{t}{\tau}} + p_b + R \int_{\Gamma_{\text{outlet}}^h} \mathbf{u}^h(t) \cdot \mathbf{n} d\Gamma \right. \right. \\ & \quad \left. \left. + \int_0^t \frac{e^{-(t-s)/\tau}}{C} \left(\int_{\Gamma_{\text{outlet}}^h} \mathbf{u}^h(s) \cdot \mathbf{n} d\Gamma \right) ds \right) \mathbf{I} + \mu \nabla \mathbf{u}^h \right) \cdot \Phi^h \cdot \mathbf{n} d\Gamma \\ & \quad - \underbrace{\sum_{K \in \mathcal{T}^h} \left(\frac{\partial \mathbf{u}^h}{\partial t} + (\mathbf{u}^h \cdot \nabla) \mathbf{u}^h + \nabla p^h, \tau_m (\mathbf{u}^h \cdot \nabla \Phi^h + \nabla \varphi^h) \right)}_K \\ & \quad - \underbrace{\sum_{K \in \mathcal{T}^h} (\nabla \cdot \mathbf{u}^h, \tau_c \nabla \cdot \Phi^h)}_K, \\ & \quad \int_{\Omega} (\nabla \cdot \mathbf{u}^h) \varphi^h d\Omega = 0 \end{aligned} \right. \quad (5)$$

holds for all $\Phi^h \in \mathcal{U}_0^h$ and $\varphi^h \in \mathcal{P}^h$. The underlined terms are the stabilization terms with parameters τ_c and τ_m defined as:

$$\begin{aligned}\tau_m &= \left(\sqrt{\frac{4}{\Delta t^2} + (\mathbf{u}^h \cdot \mathbf{G} \mathbf{u}^h) + 36 \left(\frac{\mu}{\rho} \right)^2 \mathbf{G} : \mathbf{G}} \right)^{-1}, \\ \tau_c &= \frac{1}{8\tau_m \text{tr}(\mathbf{G})}.\end{aligned}$$

Here \mathbf{G} is the covariant metric tensor whose components are defined as $G_{ij} = \sum_{k=1}^3 \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j}$ ($i, j = 1, 2, 3$) and $\frac{\partial \xi}{\partial \mathbf{x}}$ represents the Jacobian of the mapping between the reference and the physical element. $\text{tr}(\mathbf{G})$ is the trace of the matrix \mathbf{G} . The operator “ \cdot ” is the double inner product defined as $\mathbf{G} : \mathbf{G} = \sum_{i=1}^3 \sum_{j=1}^3 G_{ij} G_{ij}$.

For the temporal discretization, a second-order backward differentiation formula (BDF2)⁴¹ is applied. For the semi-discretized system (5).

$$\frac{d\mathbf{X}}{dt} = \mathbf{L}(\mathbf{X}), \quad (6)$$

the formula is defined as

$$\frac{\mathbf{X}^n - \frac{4}{3}\mathbf{X}^{n-1} + \frac{1}{3}\mathbf{X}^{n-2}}{\Delta t} = \frac{2}{3}\mathbf{L}(\mathbf{X}^n).$$

BDF2 requires two initial conditions \mathbf{X}^0 and \mathbf{X}^1 . \mathbf{X}^0 is given, and \mathbf{X}^1 is computed from (6) using the backward Euler method (It is a first order method which only needs one initial condition) with a smaller time step size $\Delta t/2$. Because of the implicitness of the method, a large nonlinear algebraic system needs to be solved at each time step and a parallel solver for the system will be introduced in the next section.

3 | A SCALABLE PARALLEL SOLVER

Let us denote the nonlinear system at the n th time step as

$$\mathbf{F}^n(\mathbf{X}^n) = \mathbf{0}, \quad (7)$$

where \mathbf{X}^n is the vector consisting of the nodal values of the velocity and pressure. A Newton-Krylov-Schwarz (NKS) method is used to solve the nonlinear system (7). A framework of NKS is shown in Algorithm 1, which has three main components, an inexact Newton method with a line search method is used to handle the nonlinear system, a Krylov subspace method is introduced to solve the Jacobian system at each Newton step (since the Jacobian system is non-symmetric, GMRES⁴² is used here), and an overlapping Schwarz method is employed as the preconditioner to accelerate the Krylov method. The Schwarz preconditioner is necessary because the condition number of the Jacobian system is usually very large, especially when the mesh is very fine. In NKS, the most time consuming step is the solution of the Jacobian system whose conditioning depends on many factors including the spatial mesh size, the temporal step size, the complexity of the fluid flow, the geometry of the computational domain, the boundary conditions, and also the physical parameters. A strong preconditioner is needed to conquer the badness of the Jacobian matrix so that the Krylov iteration converges in a reasonable number of steps. Many Schwarz type preconditioners are available, but their coarse component is either too difficult to construct or too large to be scalable for the problem at hand, in this paper we design a coarse problem suitable for the cerebral flow problems.

Algorithm 1 Newton-Krylov-Schwarz method

Step 1. Use the solution of the previous time step as the initial guess $\mathbf{X}_0^n = \mathbf{X}^{n-1}$.

Step 2. For $k = 0, 1, \dots$ until convergence.

- Construct the Jacobian matrix \mathbf{J}_k^n of $\mathbf{F}^n(\mathbf{X})$ at \mathbf{X}_k^n .
- Solve the following right-preconditioned Jacobian system.

inexactly by a Krylov subspace method

$$\mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} \mathbf{M}_k^n \mathbf{d}_k^n = -\mathbf{F}^n(\mathbf{X}_k^n) \quad (8)$$

- Do a line search to find a step length τ_k^n .
 - Set $\mathbf{X}_{k+1}^n = \mathbf{X}_k^n + \tau_k^n \mathbf{d}_k^n$
-

Here \mathbf{M}_k^n is a Schwarz preconditioner to be introduced shortly. The inexactness mentioned in Step 2 means that the solution of the Jacobian system (8) satisfies

$$\| \mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} \mathbf{M}_k^n \mathbf{S}_k^n + \mathbf{F}^n(\mathbf{X}_k^n) \| \leq \eta_k^n \| \mathbf{F}^n(\mathbf{X}_k^n) \|,$$

where $\eta_k^n > 0$ is the relative tolerance for the linear solver.

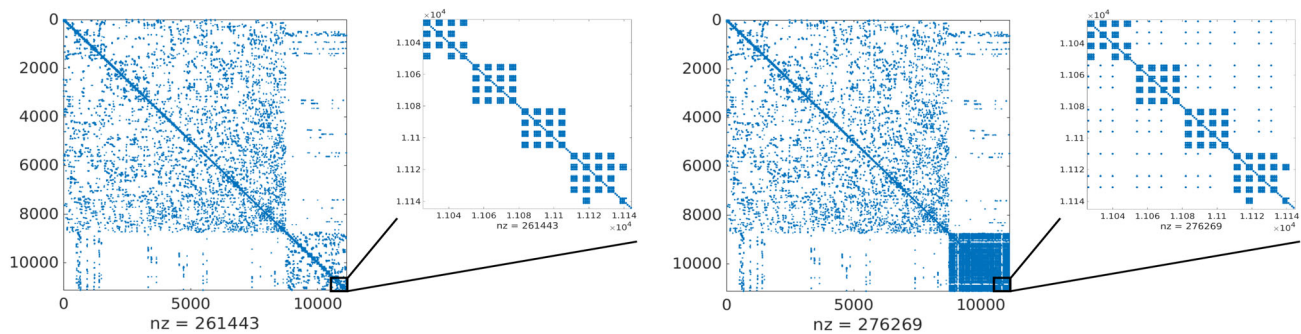
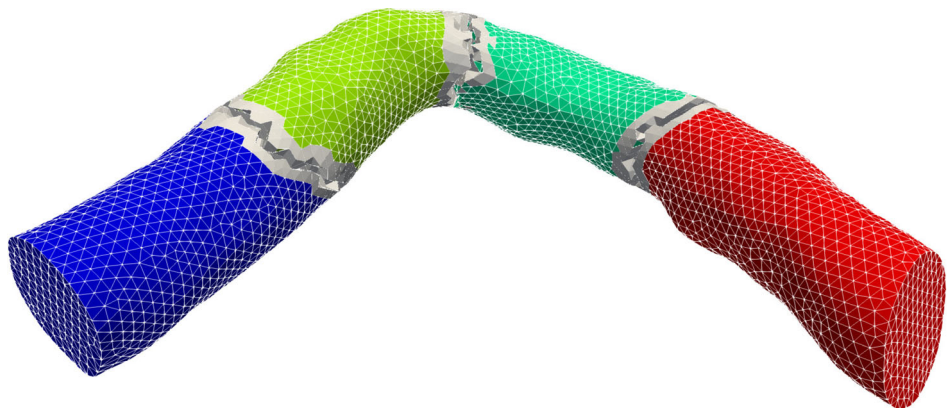


FIGURE 2 An example of the nonzero structure of the Jacobian matrices for stress-free (left) and impedance (right) boundary conditions, respectively

FIGURE 3 A sample partition of the artery into four subdomains with different colors. The gray mesh cells refer to the overlap



One particular issue with Equation (1)-(2) is the impedance outlet boundary condition which involves an integration over the boundary and time. The integration makes the Jacobian system much more difficult to solve. The nonzero structure of the Jacobian matrices with respect to the stress-free and impedance boundary conditions are shown in Figure 2, where we can see clearly that there is a relatively dense block in the matrix for the impedance boundary condition case.

To introduce the preconditioner, we denote the Jacobian system at each Newton step as

$$\mathbf{Ax} = \mathbf{b}.$$

Note that the Jacobian matrix \mathbf{A} can be calculated by finite difference method or analytically. In this paper, we use the analytic version which is very time consuming in terms of the programmer's time, but offer better performance and robustness. The overlapping restricted additive Schwarz method begins with a partition of the computational mesh into n_p nonoverlapping subdomains Ω_k , ($k = 1, \dots, n_p$), and then extend them into overlapping subdomains Ω_k^δ , ($k = 1, \dots, n_p$) by including δ layers of mesh cells from the neighboring subdomains; see Figure 3 for example. Then we build the matrix \mathbf{A}_k , ($k = 1, \dots, n_p$) on each subdomain in a similar way as the construction of the global matrix \mathbf{A} . The one-level restricted additive Schwarz preconditioner is defined as

$$\mathbf{M}_{\text{one}}^{-1} = \sum_{k=1}^{n_p} (R_k^0)^T \mathbf{B}_k^{-1} R_k^\delta,$$

where R_k^0 and R_k^δ are restriction operators from the global domain Ω to the nonoverlapping subdomain Ω_k and overlapping subdomain Ω_k^δ , respectively. \mathbf{B}_k^{-1} is a subdomain preconditioner for \mathbf{A}_k and its product with a vector is computed approximately by solving a subdomain linear system using a point-block ILU factorization.

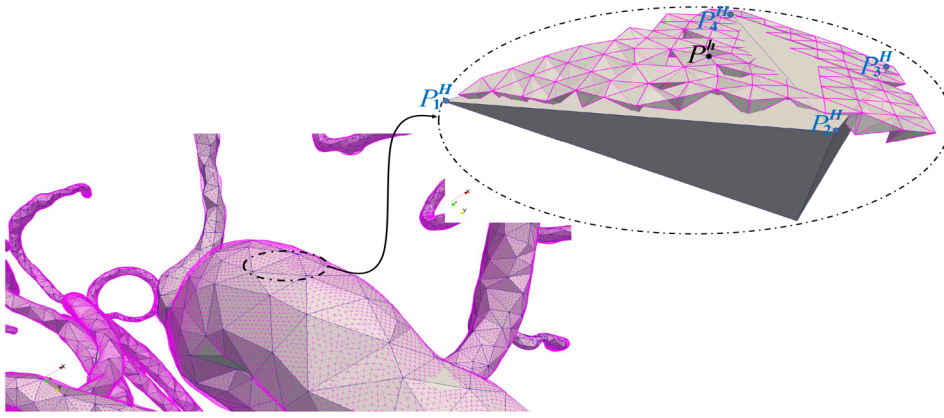


FIGURE 4 A sample non-nested fine (pink) and coarse (blue) meshes. In the zoomin figure, the value of the function at the fine mesh point P^h is obtained by RBF centered at the coarse mesh points P_i^H , $i = 1, 2, \dots, 4$. The parameter ϵ is chosen as the inscribed sphere of the polyhedron based on the points P_i^H , $i = 1, 2, \dots, 4$

The point-block ILU method is based on the point-by-point ordering of the Jacobian matrix, which means that all the unknowns \mathbf{u} and p that belong to one mesh point are ordered together. The nonzero structure of the point-by-point ordering matrix is shown in Figure 2, which is organized as 4×4 blocks (at each mesh point, there are 3 velocity components and 1 pressure). During the point-block ILU factorization, each 4×4 block is treated as one component and the corresponding matrix is inverted analytically.

There are many ways to incorporate a coarse preconditioner into a one-level method, such as additive and multiplicative methods, here we consider a hybrid Schwarz preconditioner

$$\mathbf{M}_{\text{two}}^{-1} = \mathbf{I}_H^h \mathbf{B}_c^{-1} (\mathbf{I}_H^h)^T + \sum_{k=1}^{n_p} (\mathbf{R}_k^0)^T \mathbf{B}_k^{-1} \mathbf{R}_k^\delta \left(\mathbf{I} - \mathbf{A} \mathbf{I}_H^h \mathbf{B}_c^{-1} (\mathbf{I}_H^h)^T \right),$$

where \mathbf{I}_H^h is an interpolation matrix from the coarse mesh to the fine mesh and \mathbf{B}_c^{-1} is a coarse-level preconditioner which is an approximation of the inverse of \mathbf{A}_c .

There are several ways to obtain \mathbf{A}_c , such as the Galerkin projection $\mathbf{A}_c = (\mathbf{I}_H^h)^T \mathbf{A} \mathbf{I}_H^h$. In this paper, we construct \mathbf{A}_c directly by discretizing the original problem on the coarse mesh. In the two-level method, the key questions are: (a) how to choose a good coarse mesh? (b) how to efficiently solve the coarse problem? (c) how to interpolate the coarse solution to the fine mesh and how to restrict the fine mesh solution to the coarse mesh? In most two-level methods,^{43,30} the fine mesh and coarse mesh are nested. The advantage of the nested approach is that the construction of the restriction and interpolation matrices are simple and comes at nearly no cost, but the disadvantage is that when the geometry of the artery becomes complicated the coarsening algorithm may not work well. For the patient-specific cerebral artery, to obtain a nested coarse mesh from the fine mesh is quite difficult because of the complex geometry. In our two-level method, we generate the coarse and fine meshes independently.

3.1 | The construction of the interpolation/restriction operators

Because the geometry represented by the coarse and fine meshes are not exactly the same, the construction of the interpolation and restriction operators between the coarse and fine meshes is nontrivial. With a pair of non-nested unstructured meshes it may happen that a fine mesh point is not contained in any coarse elements and a coarse mesh point may also not be in any fine elements; see Figure 4 for example. In these situations, the standard finite element based interpolation and restriction do not work. To deal with non-nested meshes, we propose to use a meshless method based on the radial basis function (RBF)⁴⁴ which works well for the situation when some points are outside of the computational domain defined by another mesh.

We now describe how the coarse to fine mesh interpolation operator is constructed. The RBF interpolation begins with finding several nearest coarse mesh points, for example, in Figure 4, to define a value at the fine mesh point P^h , we first search for the nearest points from the coarse mesh, for example, P_1^H , P_2^H , P_3^H , and P_4^H . For each of the coarse mesh points, we define a radial basis function, for example, in this paper, we use the multiquadric function

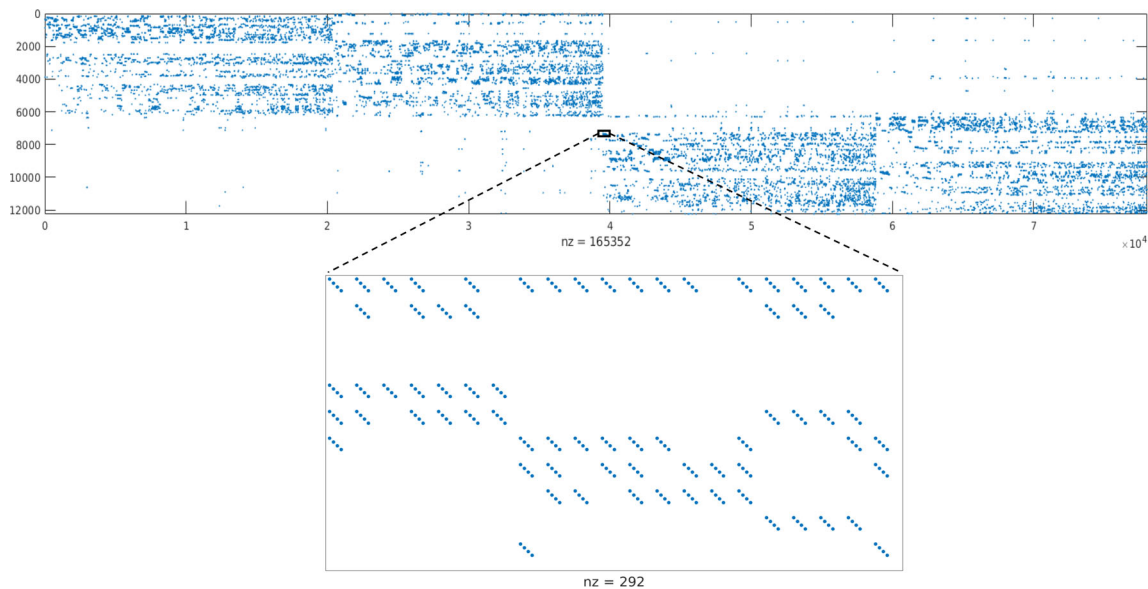


FIGURE 5 An example of the nonzero structure of the restriction matrix $(I_H^h)^T$

$$\phi_{P_i^h}(r) = \sqrt{1 + (\varepsilon r)^2},$$

where $r = \|P^h - P_i^h\|$ is the distance of the i th interpolation point P_i^h and the point that to be interpolated P^h . ε is a control parameter and is chosen as $1/H$, where H is the diameter of the inscribed sphere of the polyhedron based on the interpolation points, for example, P_1^h, P_2^h, P_3^h , and P_4^h in Figure 4. Let n_h be the number of fine mesh points and n_H the coarse mesh points, then the coarse to fine interpolation matrix I_H^h is $4n_h \times 4n_H$. Note that there are multiple field values associated with each mesh point, and they all share the same interpolation. I_H^h is partitioned into n_{cp} (the number of partitions of the coarse mesh) submatrices and each processor corresponding to a coarse mesh partition has one of the submatrices. Figure 5 gives an example of the nonzero structure of $(I_H^h)^T$. In the matrix, each column has 4 nonzero elements, corresponding to the 4 nodes used for the interpolation. Since we use the point-block ordering for the matrix, all elements corresponding to the variables defined at a mesh point are grouped together as showed in the zoom-in part of the figure (at each mesh point, there are 3 velocity and 1 pressure components).

The same RBF method can be used to define the restriction operator since the method does not distinguish whether the computation is from the coarse mesh to the fine mesh or vice versa. In this paper, the restriction operator is chosen as the transpose of the interpolation operator.

3.2 | Remark

The number of coarse grid points needed in the interpolation depends on the required accuracy that is often determined experimentally based on the problems under consideration. In this paper, we use 4 coarse mesh points because when the fine mesh point is in a coarse element, the 4 vertices will be in the same element. This provides better performance than using other points outside the element as the communication cost may increase. For each fine mesh point, we loop through all the coarse mesh points and find the nearest 4 points. In order to reduce the cost of communication, we let each processor have a full copy of the coarse mesh. The interpolation matrix is constructed offline and then loaded to the machine once the computation starts. This increases the I/O time, but decreases the communication time.

3.3 | Partitioning and solving the coarse problem

The coarse mesh is much smaller than the fine mesh with a ratio between 50 to 100 for problems considered in this paper. It is not a good idea to use the same number of processors for the fine and coarse problems especially when the

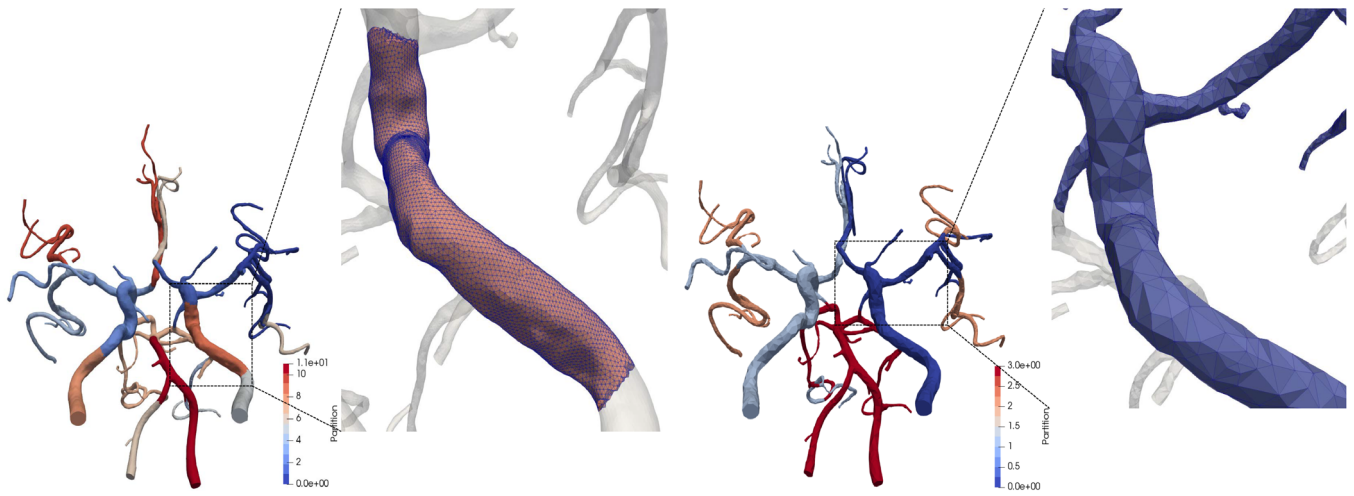


FIGURE 6 An example of the a fine mesh (left) partitioned into 12 subdomains, and a coarse mesh (right) of the same artery partitioned into 4 subdomains. The meshes are not nested, and the partitions are obtained independently. Different color refers to different subdomains

number of processors is large. In this paper, we use a smaller number of processors for the coarse problem. The partition of the coarse and fine meshes are obtained independently by ParMETIS⁴⁵; see Figure 6 for an example of the partition. We remark that we use a subset of the fine partition processors to solve the coarse problem and let the other processors stay idle. This may seem like a waste of resources, but it actually saves a lot of communication time. For the coarse problem, the one-level additive Schwarz preconditioned GMRES method is used as the solver, where we use the point-block ILU method with one level of fill-in as the subdomain solver, the overlapping size of the additive Schwarz method is set to one, and use the maximum iteration number (40) as the stopping condition for GMRES.

4 | NUMERICAL EXPERIMENTS

In this section we present some numerical experiments to validate the proposed algorithm, its implementation, and also show its robustness and efficiency. The algorithm is implemented on top of PETSc.⁴⁶ The first experiment is for the verification and validation of the algorithm and its software implementation, the second set of experiments focuses on the physics of the fluids of a cerebral artery with stenosis, and the third set of experiments is to show the numerical and parallel performance of the algorithm.

4.1 | A comparison of computed and clinically measured results

We consider blood flows in a vertebrobasilar artery with a 90% area reduction stenosis. For this particular patient, the pressure field is measured by a pressure wire at two points, namely the front-end Pa and the back-end Pd of the stenosis (see Figure 7 for the detail of the artery and the positions of Pa and Pd), as shown in.⁴⁷ In the numerical experiment, the total resistance R_{total} and capacitance C_{total} are chosen such that the mean pressure at the inlet boundary matches with the measured pressure, that is $R_{\text{total}} = 5 \times 10^5 \text{ dyne} \cdot \text{s/cm}^5$ and $C_{\text{total}} = 1.0 \times 10^1$. The patient-specific inflow rate for the inlet boundary is shown in the left figure of Figure 7. A finite element mesh with 1.5×10^6 elements is used and the time step size is 0.01s for the entire cardiac cycle. The results are shown in Figure 8. The computed pressure matches with the measured pressure well at Pa but there is a small discrepancy at Pd. The possible explanation is that the pressure wire disturbs slightly the downstream flow because the diameter of the artery at the stenosis is about 1.3 mm and the diameter of the pressure wire is about 0.36 mm, which means 8% of the area at the stenosis is blocked by the pressure wire, but in the simulation, the impact of the pressure wire is not included. The averaged ratios of the pressure values at Pd and Pa over the cardiac cycle (which is an important parameter to indicate the lack of blood supply for the downstream area of the brain) obtained from the measured and the computed pressure are 0.8756 and 0.882, respectively, that are considered the same for clinical purpose.

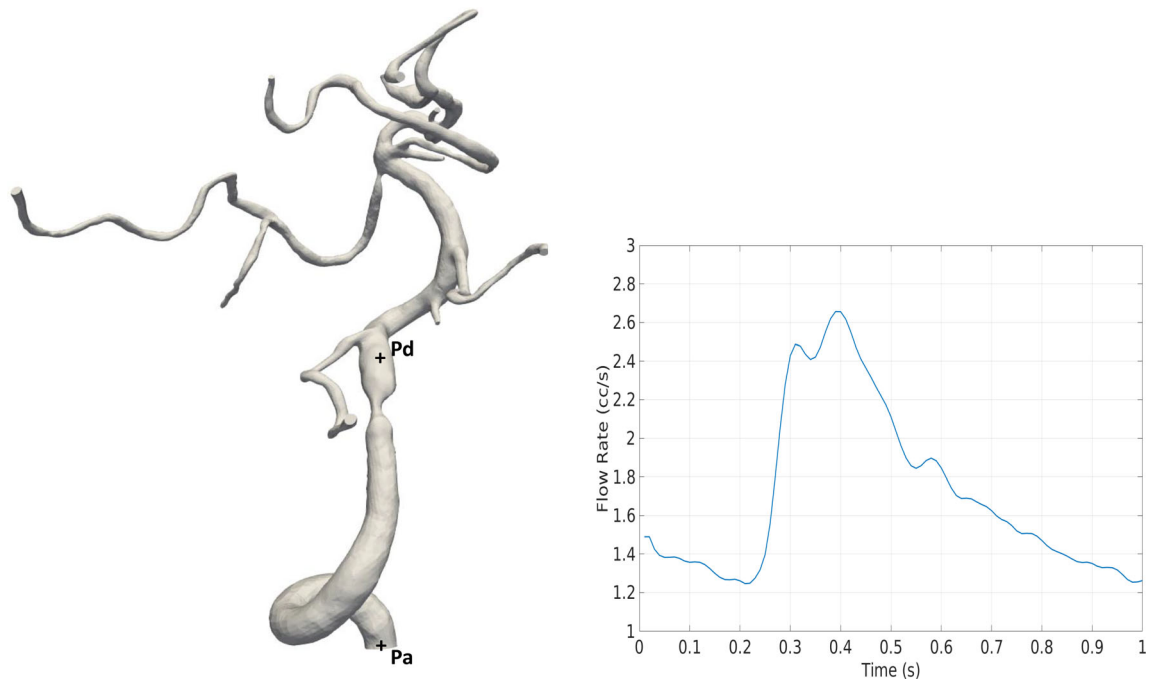


FIGURE 7 The geometry of the vertebrobasilar artery (left) and the patient-specific flow rate at the inlet boundary (right). Pa and Pd are the two points where the pressure is measured by the pressure wire

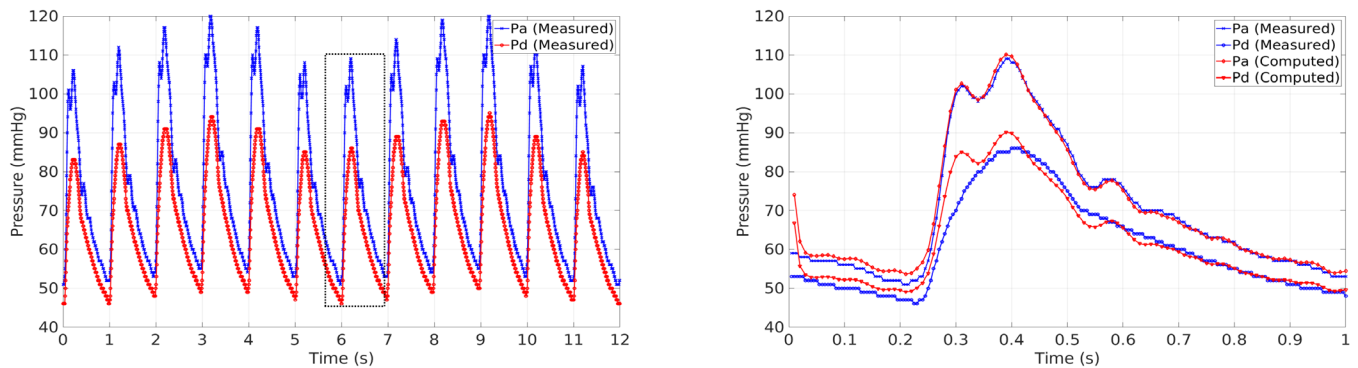


FIGURE 8 The left figure is the measured pressures at two points Pa and Pd (see Figure 7 for the position of Pa and Pd) for 12 cardiac cycles. The right figure is the comparison of the measured and computed pressures at these two points for one of the cardiac cycle (the boxed cycle in the left figure)

4.2 | Flow in a full cerebral artery with stenosis

In this section, we compute the blood flow of a full size cerebral artery which has a stenosis in the left middle part of the artery and the blockage is about 55% in terms of the cross section area; see Figure 9 for details. The computational domain and the boundary conditions are shown in Figure 1. The artery has four inlet boundaries and 38 outlet boundaries. On the inlet boundaries, a patient-specific flow rate is given; see Figure 10. For this computation, we use a finite element mesh with 1.1×10^7 elements and the time step size $\Delta t = 0.01$ (sec) for the entire cardiac cycle (1.0 second). The average mesh size is around 0.02 mm, and the resolution is higher than MRI which is around 0.5 mm. The total resistance R_{total} and capacitance C_{total} in the impedance boundary condition are 3.0×10^4 dyne \cdot s/cm⁵ and 2.5×10^{-5} cm⁵/dyne, respectively. The stopping condition for the linear and nonlinear solvers are 10^{-3} and 10^{-6} (relative condition), respectively.

The computed pressure distributions at the systolic stage ($t = 0.2$ s) for the impedance and stress-free boundary conditions are shown in Figure 11 and the fractional pressure ratio (FPR) distribution is shown in Figure 12. FPR is a parameter to show the drop of the pressure defined as

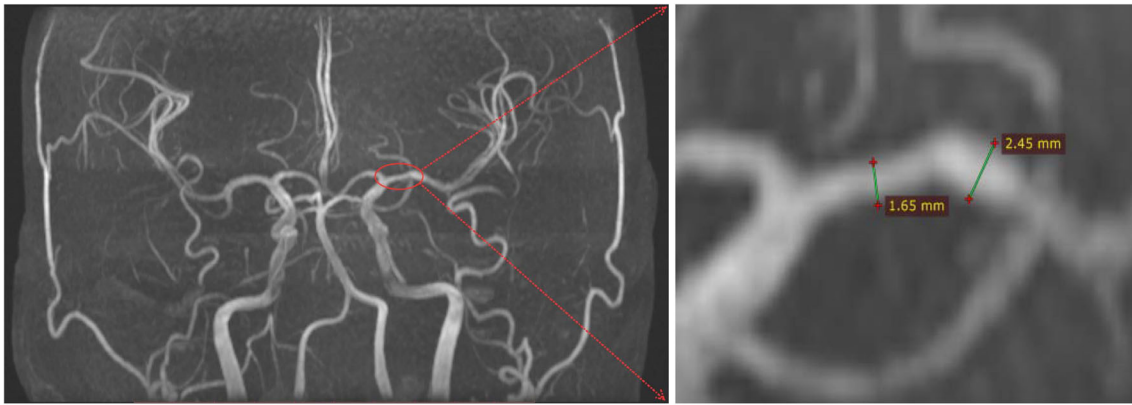


FIGURE 9 The MR medical image of a stroke case. The circled area has a 55% area reduction stenosis

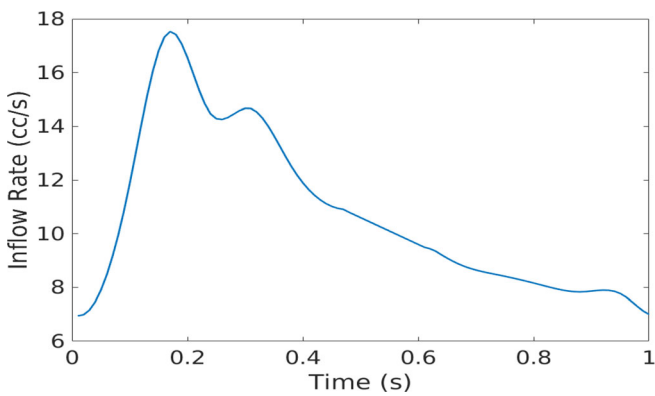


FIGURE 10 The inflow rate at the inlet boundaries

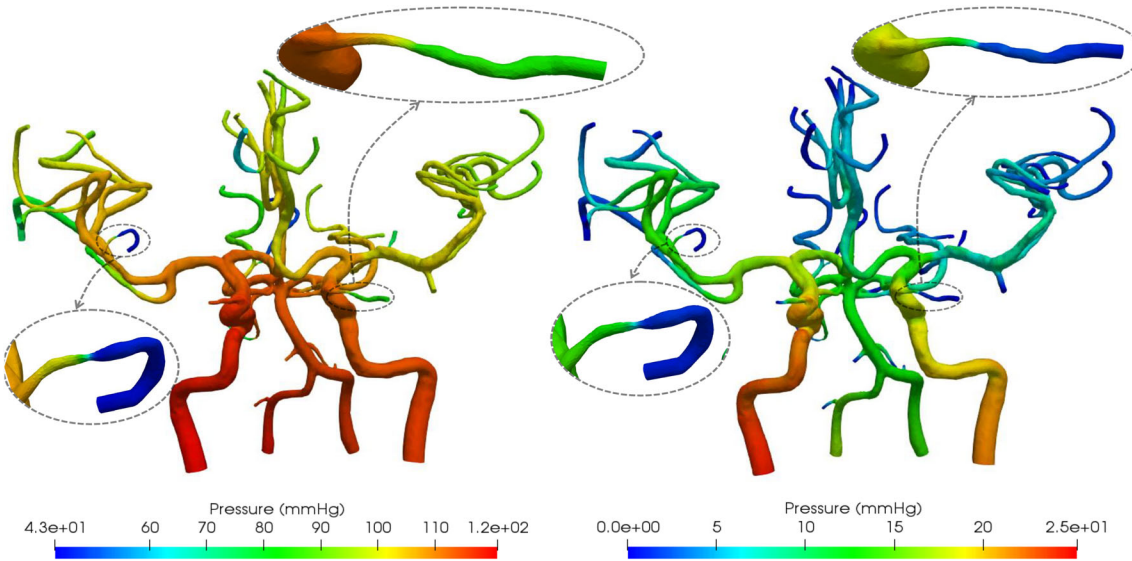


FIGURE 11 The pressure distribution at the systolic stage ($t = 0.2s$) with different outlet boundary conditions. The left figure is computed with the impedance boundary and the right figure is computed with the stress-free boundary condition

$$FPR = \frac{p}{\max_{\Omega} p},$$

that is the ratio of the pointwise pressure and its maximum over the computational domain at a given time.¹⁵ From the pressure and FPR distributions, we observe that there are two areas with obvious pressure drops at the posterior

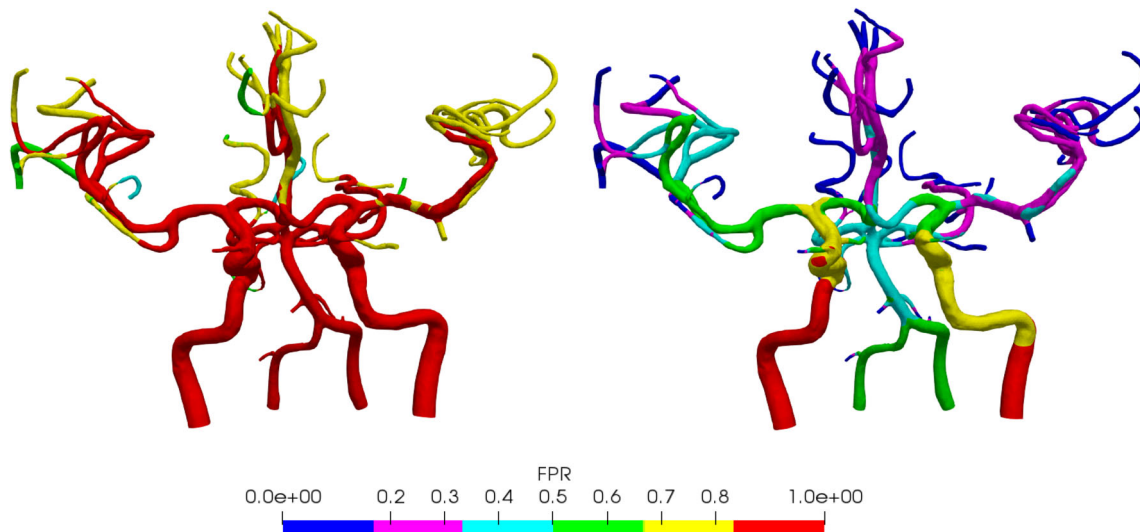


FIGURE 12 The FPR distribution at the systolic stage ($t = 0.2s$) with impedance (left) and stress-free (right) outlet boundary conditions

cerebral arteries. This is an indication that the downstream blood supply is not sufficient. Detecting the locations of the stenosis is very important and it is usually very difficult to do by just looking at the images (such as Figure 9), especially in small arteries. On the other hand, the computed pressure drop can be used to identify these stenosis easily. Moreover, there are growing evidences suggesting that the pressure drop is a better indication than the area reduction about the severity of the disease, that is if a functional measurement, such as FPR, shows that the flow is not significantly blocked (eg, $FPR > 0.8$), the blockage or lesion does not need to be surgically repaired (angioplasty/stenting), and the medical therapy is a safe treatment for the patient.^{48,47} The computed pressure with the impedance boundary condition (43 ~ 120 mmHg) is higher than the one computed with the stress-free boundary condition (0 ~ 25 mmHg). From the pressure distribution, we see that the results of the impedance boundary condition is more reasonable than the stress-free boundary condition because the pressure is too low (the systolic pressure should be higher than 90 mmHg) and the pressure drop of the arteries without stenosis is too large ($FPR < 0.5$) for the stress-free boundary condition case (FPR should be close to 1 for arteries without stenosis).^{15,47} For the rest of the experiment, we only consider the impedance boundary condition.

The wall shear stress (WSS) is an important parameter, which is believed to be related to the formation of the plaques and also the stability of existing plaques. The definition of pointwise WSS and spatially averaged wall shear stress (\overline{WSS}) are

$$WSS = \sigma \mathbf{n} - (\sigma \mathbf{n} \cdot \mathbf{n}) \mathbf{n} \quad \text{and} \quad \overline{WSS} = \frac{1}{A_{\text{wall}}} \int_{\Gamma_{\text{wall}}} WSS d\Gamma.$$

Here $\sigma = -p\mathbf{I} + 2\mu\epsilon(\mathbf{u})$ is the Cauchy stress tensor where \mathbf{I} is an 3×3 identity matrix and $\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)$.⁴⁹ A_{wall} is the total area of the artery wall. Figure 13 shows that the WSS near the stenosis is higher than other areas, which means that if the plaque is unstable it may break away easily because of the large WSS. Figure 14 shows the time-dependent wall shear stress for some branches. The values are different for different branches, but they are all correlated to the inflow rate, and this may deserve further study that is beyond the scope of this paper. The time-dependent spatially averaged wall shear stress \overline{WSS} is shown in Figure 15 which has a similar profile as the inflow rate (Figure 10). From this observation, we conclude that one can control the \overline{WSS} by controlling the flow rate at the inlets.

The velocity and streamline distributions are shown in Figures 16-17. We observe that the velocity is relatively high in the stenosis areas and the velocity in the middle of the artery is higher than the area that is near the artery wall and the distribution is not symmetric. The streamlines show that the flow is screwed in the artery and some vortices are generated due to the narrowing or non-smoothness of the artery geometry. Our computation is able to capture many details of the dynamics of the blood flow that is difficult to capture with 4D MRI.

FIGURE 13 The wall shear stress distribution

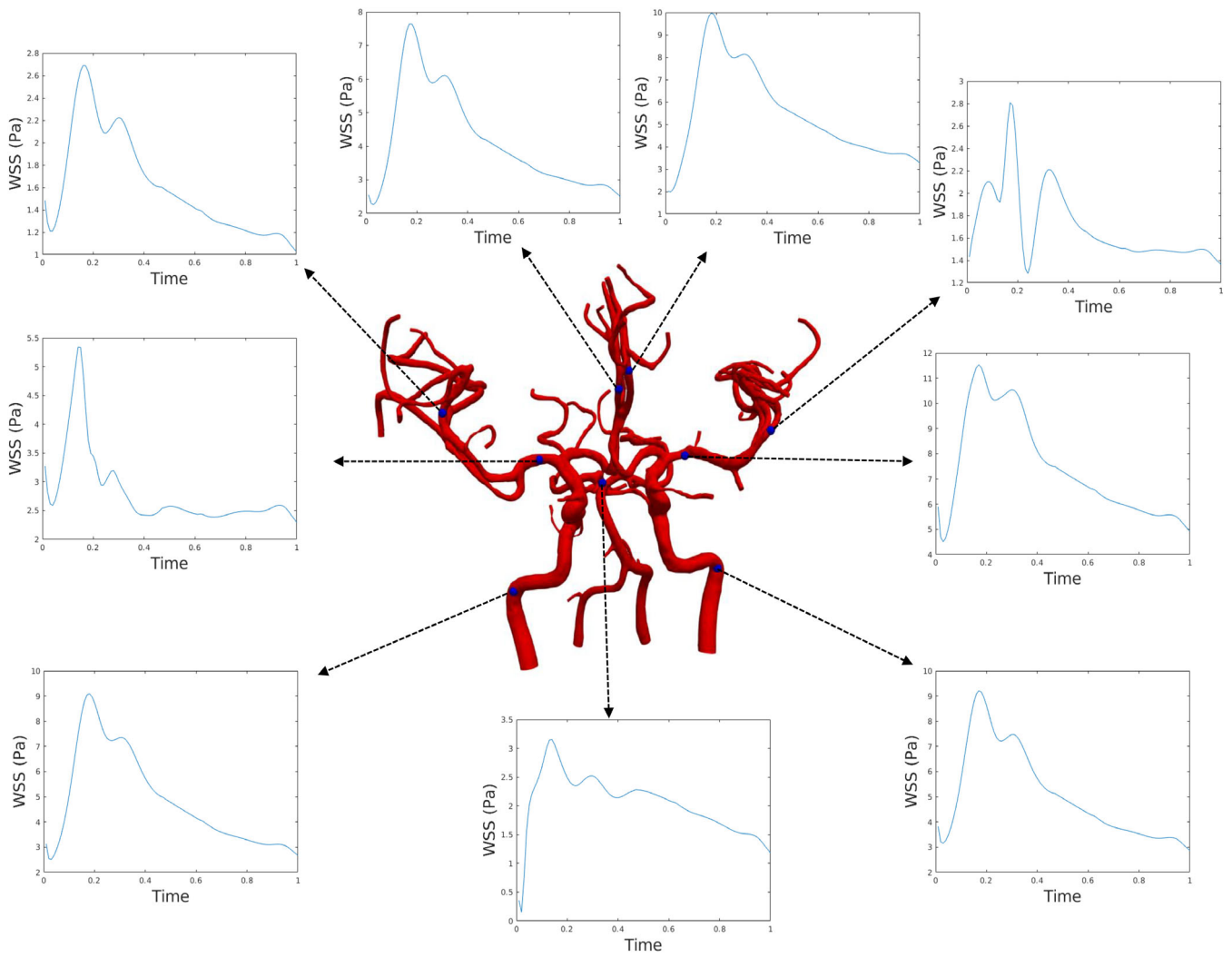
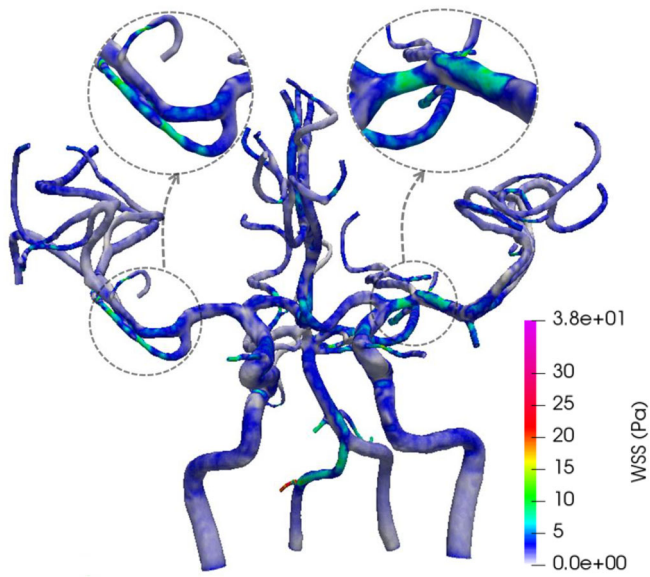


FIGURE 14 The time-dependent pointwise wall shear stress at certain locations for a cardiac cycle

FIGURE 15 The spatially averaged wall shear stress for a cardiac cycle

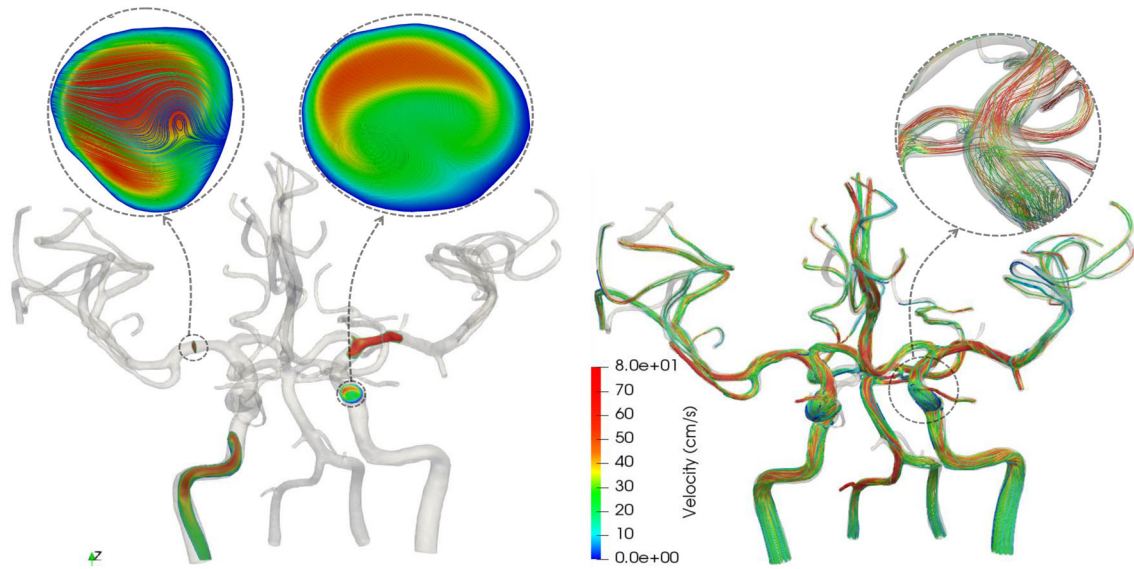
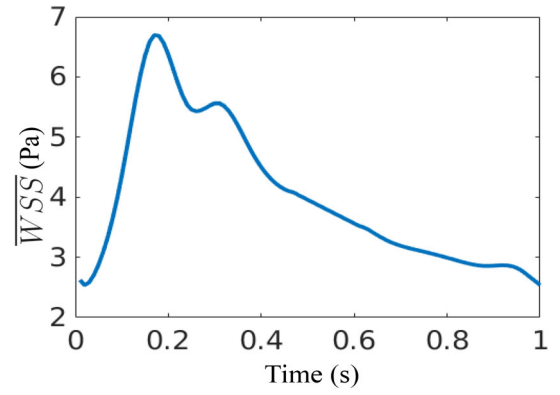


FIGURE 16 The velocity (left) distribution with some cross section views and the streamline (right) distribution with a local zoomin view. In the left figure, the lines on the two cross sections are 2D streamlines

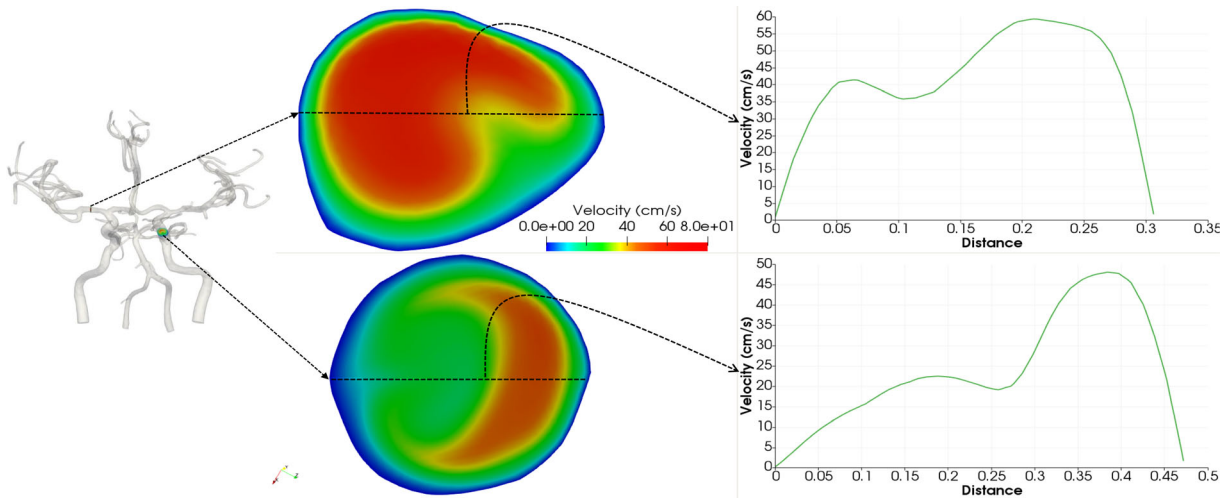


FIGURE 17 The velocity (left) distribution with two cross section views and the velocity magnitude value on the cross lines (right)

n_p	Stress-free BC			Impedance BC		
	N-Its	L-Its	Time	N-Its	L-Its	Time
One-level method						
240	3	283	166.2	4	624	338.2
280	3	328	98.2	4	722	202.5
960	3	375	61.6	4	902	129.4
1920	3	447	41.1	4	1254	98.5
Two-level method						
240	3	36	75.5	4	61	167.5
480	3	29	37.9	4	34	51.8
960	3	34	24.9	4	41	34.0
1920	3	45	20.1	4	50	28.2

TABLE 1 Comparison of the one-level and two-level methods. The number of elements for the fine and coarse meshes are 1.1×10^7 and 8.0×10^4 , respectively

4.3 | The performance study

In this section, we show the performance of the algorithm for the case studied in Section 4.1. The experiments are carried out on the Tianhe 2A supercomputer at the China National Supercomputer Center in Guangzhou. Each compute node has two Intel Xeon E5-2692v2 12-core 2.2GHz processors and 64GB of shared memory. To measure the parallel performance, we use the speedup and parallel efficiency defined as⁵⁰

$$\text{speedup} = \frac{t_M}{t_N} \text{ and efficiency} = \frac{M \times t_M}{N \times t_N},$$

where t_M and t_N are the total compute time using M and N MPI processes ($M \leq N$), respectively. M is the smallest number of processor cores that can solve the problem. In all the tables, “N-Its” and “L-Its” are the average number of Newton iterations per time step and the average number of GMRES iterations per Newton iteration. “Time” and “PC” are the average total compute time and the time spent on the preconditioner in seconds per time step. “Eff” is the efficiency. “Mem(Mb)” is the average memory usage per processor core in Megabyte. Unlike the linear system on the fine mesh, the accuracy of the coarse linear system is very loose and we stop the coarse GMRES iteration by setting a maximum number of iterations. “Coarse Its” refers to the maximum number of GMRES iterations used for the coarse problem. “ n_p ” and “ n_{cp} ” are the numbers of processor cores used for solving the fine and coarse problems, respectively.

First, we compare the performance of the one-level method and a two-level method in Table 1. In this experiment, the fine mesh has 1.1×10^7 elements, the overlap size is 2, the ILU fill-in level is 2. For the coarse problem in the two-level method, the mesh has 8.0×10^4 elements, the parameters for the fine-level solver is the same as the one-level method and for the coarse-level solver, the overlap size is 1, the ILU fill-in level is 1 and Coarse Its is 40. The comparison is carried out for 10 time steps, and the results show that the two-level method is faster in terms of the number of iterations and also the total compute time. For the stress-free boundary condition, the saving is about 1/2, and for the impedance boundary condition it is about 2/3. By comparing the number of linear and nonlinear iterations, we see clearly that the impedance boundary condition case is much more difficult to solve than the stress-free boundary condition case. Table 2 shows that with the increase of Coarse Its, the number of GMRES iterations decreases, which means the preconditioner is stronger, however the total compute time first decreases and then increases. The best choice of Coarse Its is 40 for the case in Table 2 for the optimal compute time.

As pointed out in Section 3, we do not need to use the same number of processor cores for the fine and coarse problems, since the coarse problem is much smaller than the fine problem (the ratio of the problem sizes is between 50 to 100). Table 3 shows the performance of the algorithm using different number of processor cores for solving the coarse problem when the number of processor cores for the fine mesh problem is fixed to 1200. We observe that when n_{cp} increases from 600 to 1200, the number of GMRES iterations increases from 29 to 83. The reason is that the increase of n_{cp} slows down the one-level additive Schwarz preconditioned GMRES method for the coarse problem. Since we fix Coarse Its to be 40 for all cases, the accuracy of the coarse problem solution decreases and the global preconditioner becomes weaker. The best choice of n_{cp} is 600 for this case in Table 3.

TABLE 2 Comparison using different number of GMRES iterations for the coarse problem. The number of processor cores is 480

Coarse Its	N-Its	L-Its	Time
20	4	34	50.1
30	4	28	45.6
40	4	25	45.1
50	4	26	46.0
60	4	25	48.7

Note: The number of elements for the fine and coarse meshes are 1.1×10^7 and 8.0×10^4 , respectively.

TABLE 3 Comparison of different number of processor cores allocated for the coarse problem

n_{cp}	N-Its	L-Its	Time
75	4	29	32.4
150	4	29	27.6
300	4	29	25.0
600	4	29	24.6
1200	4	83	39.7

Note: The number of processor cores for the fine mesh problem is fixed at 1200. The number of elements for the fine and coarse meshes are 1.1×10^7 and 8.0×10^4 , respectively.

TABLE 4 Parallel performance

n_p	One-level						Two-level					
	N-Its	L-Its	Time	Eff	PC	Mem (Mb)	N-Its	L-Its	Time	Eff	PC	Mem (Mb)
Mesh 1: Number of Elements = 1.9×10^7												
480	4	373	178.6	100%	64.1	841.0	4	50	73.1	100%	52.4	1186.9
960	4	441	118.5	75.4%	44.6	439.5	4	58	45.1	81.0%	30.6	611.6
1920	4	536	95.7	46.7%	37.6	240.2	4	70	34.0	53.8%	23.5	331.0
3840	4	691	84.6	26.4%	33.4	123.2	4	76	19.2	47.6%	17.1	175.3
Mesh 2: Number of Elements = 2.6×10^7												
480	4	393	259.7	100%	91.7	1144.5	4	57	101.2	100%	72.9	1534.1
960	4	474	167.6	77.5%	62.4	597.6	4	67	55.5	91.2%	41.4	802.5
1920	4	569	124.4	52.2%	48.8	322.4	4	76	38.8	65.2%	30.8	440.0
3840	4	756	95.4	34.0%	38.7	163.3	4	82	22.2	57.0%	19.2	221.2

Note: The number of elements for the coarse mesh is 8.0×10^4 .

Table 4 shows the parallel performance of the one-level and two-level algorithms. We consider two meshes. The first mesh has 1.9×10^7 tetrahedral elements and the second one has 2.6×10^7 elements. The results show that with the increase of the number of processor cores, the number of Newton iterations stays a constant, the numbers of GMRES iterations for the one-level and two-level methods increase about 80% and 50%, respectively, when the number of processor cores increases from 480 to 3840, and the total compute time decreases. The efficiency of the one-level and two-level algorithms are around 34% and 57%, respectively, when the number of processor cores reaches 3840.

Figure 18 shows the speedup of the algorithm. The blue line is the ideal speedup and the green and red lines are the actual speedups for the two meshes. The speedup starts to drop when the number of processor cores is over 960 because the problem size is too small (the size of the subdomain problem is smaller than 1×10^4 and each processor core can handle about 5×10^4 unknowns) for the case with larger processor counts. But overall, the speedup is about 47% of the ideal speedup that is good for such a complicated case.

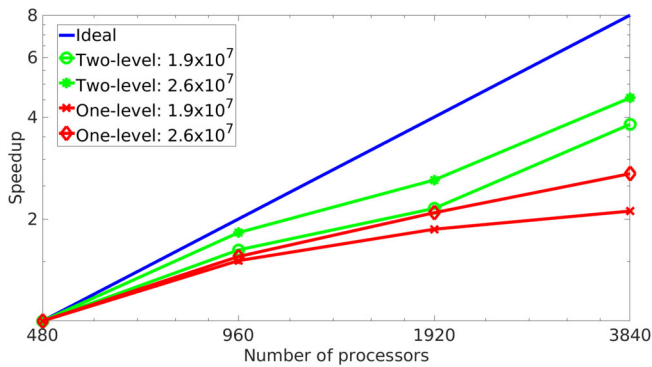


FIGURE 18 The speedup of the one-level and two-level methods

5 | FINAL REMARKS

We introduced a parallel two-level domain decomposition method with non-nested coarse space for solving three-dimensional Navier-Stokes equations for the simulation of blood flows in full size cerebral artery with an integral type boundary condition on a large scale supercomputer. We compared our computation with some clinical studies and the results agree well. The numerical experiments also show that the proposed method outperforms the one-level method in terms of the number of linear and nonlinear iterations and also the total compute time. The algorithm scales up to 3840 processor cores with a parallel efficiency higher than 47%. For patient-specific geometry and parameters, the proposed algorithm is able to carry out the flow simulation for a cardiac cycle in less than 1 hour, and the resolution is higher than 4D MRI. This takes us one step closer to the application of parallel CFD in the study of strokes and other diseases in the cerebral artery and provides a better way to identify the severity of the disease.

ACKNOWLEDGEMENTS

This research was supported by the National Key R&D Program of China under 2016YFB0200601 and Shenzhen grants under ZDSYS201703031711426 and JCYJ20170818153840322.

ORCID

Xiao-Chuan Cai  <https://orcid.org/0000-0003-0296-8640>

REFERENCES

- Cecchi E, Giglioli C, Valente S, et al. Role of hemodynamic shear stress in cardiovascular disease. *Atherosclerosis*. 2011;214(2):249-256. <https://doi.org/10.1016/j.atherosclerosis.2010.09.008>.
- Nixon AM, Gunel M, Sumpio BE. The critical role of hemodynamics in the development of cerebral vascular disease: a review. *J Neurosurg*. 2010;112(6):1240-1253. <https://doi.org/10.3171/2009.10.JNS09759>.
- Xiang J, Natarajan SK, Tremmel M, et al. Hemodynamic-morphologic discriminants for intracranial aneurysm rupture. *Stroke*. 2011;42(1):144-152. <https://doi.org/10.1161/STROKEAHA.110.592923>.
- Callaghan F, Karkouri J, Broadhouse K, Evin M, Fletcher D, Grieve S. Thoracic aortic aneurysm: 4D flow MRI and computational fluid dynamics model. *Comput Methods Biomech Biomed Eng*. 2015;18(sup1):1894-1895. <https://doi.org/10.1080/10255842.2015.1069559>.
- Stankovic Z, Allen BD, Garcia J, Jarvis KB, Markl M. 4D flow imaging with MRI. *Cardiovasc Diagn Ther*. 2014;4(2):173-192. <https://doi.org/10.3978/j.issn.2223-3652.2014.01.02>.
- Chan BT, Lim E, Chee KH, Osman NAA. Review on CFD simulation in heart with dilated cardiomyopathy and myocardial infarction. *Comput Biol Med*. 2013;43(4):377-385. <https://doi.org/10.1016/j.compbimed.2013.01.013>.
- Dedè L, Menghini F, Quarteroni A. Computational fluid dynamics of blood flow in an idealized left human heart. *Int J Numer Methods Biomed Eng*. 2019. <https://doi.org/10.1002/cnm.3287>.
- Buoso S, Manzoni A, Alkadhhi H, Plass A, Quarteroni A, Kurtcuoglu V. Reduced-order modeling of blood flow for noninvasive functional evaluation of coronary artery disease. *Biomech Model Mechanobiol*. 2019;18(6):1867-1881. <https://doi.org/10.1007/s10237-019-01182-w>.
- Stoter SK, Müller P, Cicalese L, Tuveri M, Schillinger D, Hughes TJ. A diffuse interface method for the Navier-stokes/Darcy equations: perfusion profile for a patient-specific human liver based on MRI scans. *Comput Methods Appl Mech Eng*. 2017;321:70-102. <https://doi.org/10.1016/j.cma.2017.04.002>.
- Zhong L, Zhang JM, Su B, Tan RS, Allen JC, Kassab GS. Application of patient-specific computational fluid dynamics in coronary and intra-cardiac flow simulations: challenges and opportunities. *Front Physiol*. 2018;9:742. <https://doi.org/10.3389/fphys.2018.00742>.
- Urick B, Sanders TM, Hossain SS, Zhang YJ, Hughes TJ. Review of patient-specific vascular modeling: template-based isogeometric framework and the case for CAD. *Arch Comput Methods Eng*. 2019;26(2):381-404. <https://doi.org/10.1007/s11831-017-9246-z>.

12. Campo-Deano L, Oliveira MSN, Pinho FT. A review of computational hemodynamics in middle cerebral aneurysms and rheological models for blood flow. *Appl Mech Rev*. 2015;67(3):030801. <https://doi.org/10.1115/1.4028946>.
13. Yu H, Huang GP, Yang Z, Ludwig BR. A multiscale computational modeling for cerebral blood flow with aneurysms and/or stenoses. *Int J Numer Methods Biomed Eng*. 2018;34(10):1-14. <https://doi.org/10.1002/cnm.3127>.
14. Leng X, Scalzo F, Ip HL, et al. Computational fluid dynamics modeling of symptomatic intracranial atherosclerosis may predict risk of stroke recurrence. *PLoS ONE*. 2014;9(5):e97531. <https://doi.org/10.1371/journal.pone.0097531>.
15. Liu J, Yan Z, Pu Y, et al. Functional assessment of cerebral artery stenosis: a pilot study based on computational fluid dynamics. *J Cereb Blood Flow Metab*. 2017;37(7):2567-2576. <https://doi.org/10.1177/0271678X16671321>.
16. Luo L, Shiu WS, Chen R, Cai XC. A nonlinear elimination preconditioned inexact Newton method for blood flow problems in human artery with stenosis. *J Comput Phys*. 2019;399:108926. <https://doi.org/10.1016/j.jcp.2019.108926>.
17. Liang F, Oshima M, Huang H, Liu H, Takagi S. Numerical study of cerebroarterial hemodynamic changes following carotid artery operation: a comparison between multiscale modeling and stand-alone three-dimensional modeling. *J Biomech Eng*. 2015;137(10):101011. <https://doi.org/10.1115/1.4031457>.
18. Passerini T, Luca M, Formaggia L, Quarteroni A, Veneziani A. A 3D/1D geometrical multiscale model of cerebral vasculature. *J Eng Math*. 2009;64(4):319-330. <https://doi.org/10.1007/s10665-009-9281-3>.
19. Ghaffari M, Tangen K, Alaraj A, Du X, Charbel FT, Linninger AA. Large-scale subject-specific cerebral arterial tree modeling using automated parametric mesh generation for blood flow simulation. *Comput Biol Med*. 2017;91:353-365. <https://doi.org/10.1016/J.COMPBIOMED.2017.10.028>.
20. Ghaffari M, Alaraj A, Du X, Zhou XJ, Charbel FT, Linninger AA. Quantification of near-wall hemodynamic risk factors in large-scale cerebral arterial trees. *Int J Numer Methods Biomed Eng*. 2018;34(7):e2987. <https://doi.org/10.1002/cnm.2987>.
21. Forti D, Quarteroni A, Deparis S. A parallel algorithm for the solution of large-scale nonconforming fluid-structure interaction problems in hemodynamics. *J Comput Math*. 2017;35(3):363-380. <https://doi.org/10.4208/jcm.1702-m2016-0630>.
22. Wu Y, Cai XC. A fully implicit domain decomposition based ALE framework for three-dimensional fluid-structure interaction with application in blood flow computation. *J Comput Phys*. 2014;258:524-537. <https://doi.org/10.1016/j.jcp.2013.10.046>.
23. Kong F, Cai XC. Ascalable nonlinear fluid-structure interaction solver based on a Schwarz preconditioner with isogeometric unstructured coarse spaces in 3D. *J Comput Phys*. 2017;340:498-518. <https://doi.org/10.1016/j.jcp.2017.03.043>.
24. Kong F, Cai XC. Scalability study of an implicit solver for coupled fluid-structure interaction problems on unstructured meshes in 3D. *Int J High Perform Comput Appl*. 2018;32(2):207-219. <https://doi.org/10.1002/cpe.3099>.
25. Lee S, Gounley J, Randles A, Vetter JS. Performance portability study for massively parallel computational fluid dynamics application on scalable heterogeneous architectures. *J Parallel and Distrib Comput*. 2019;129:1-13. <https://doi.org/10.1016/j.jpdc.2019.02.005>.
26. Randles A, Draeger EW, Oppelstrup T, Krauss L, Gunnels JA. *Massively parallel models of the human circulatory system*. SC'15. 2015; ACM; New York, NY. pp.1-11.
27. Chen R, Cai XC. A parallel two-level domain decomposition based one-shot method for shape optimization problems. *Int J Numer Methods Eng*. 2014;99(13):945-965. <https://doi.org/10.1002/nme.4711>.
28. Yang H, Cai XC. Two-level space-time domain decomposition methods for flow control problems. *J Sci Comput*. 2017;70(2):717-743. <https://doi.org/10.1007/s10915-016-0263-0>.
29. Barker AT, Cai XC. Two-level Newton and hybrid Schwarz preconditioners for fluid-structure interaction. *SIAM J Sci Comput*. 2010;32(4):2395-2417. <https://doi.org/10.1137/090779425>.
30. Yang C, Cai XC. Parallel multilevel methods for implicit solution of shallow water equations with nonsmooth topography on the cubed-sphere. *J Comput Phys*. 2011;230(7):2523-2539. <https://doi.org/10.1016/j.jcp.2010.12.027>.
31. Zhao T, Hwang FN, Cai XC. Parallel two-level domain decomposition based Jacobi-Davidson algorithms for pyramidal quantum dot simulation. *Comput Phys Commun*. 2016;204:74-81. <https://doi.org/10.1016/j.cpc.2016.03.009>.
32. Formaggia L, Lamponi D, Tuveri M, Veneziani A. Numerical modeling of 1D arterial networks coupled with a lumped parameters description of the heart. *Comput Methods Biomech Biomed Engin*. 2006;9(5):273-288. <https://doi.org/10.1080/10255840600857767>.
33. Moon JY, Suh DC, Lee YS, Kim YW, Lee JS. Considerations of blood properties, outlet boundary conditions and energy loss approaches in computational fluid dynamics modeling. *Neurointervention*. 2014;9(1):1-8. <https://doi.org/10.5469/neuroint.2014.9.1.1>.
34. Olufsen MS. Structured tree outflow condition for blood flow in larger systemic arteries. *Am J Physiol Heart Circul Physiol*. 1999;276(1):H257-H268. <https://doi.org/10.1152/ajpheart.1999.276.1.H257>.
35. Smith F, Purvis R, Dennis S, Jones M, Ovenden N, Tadjfar M. Fluid flow through various branching tubes. *J Eng Math*. 2003;47(3-4):277-298. <https://doi.org/10.1023/B:ENGI.0000007981.46608.73>.
36. Spilker RL, Feinstein JA, Parker DW, Reddy VM, Taylor CA. Morphometry-based impedance boundary conditions for patient-specific modeling of blood flow in pulmonary arteries. *Ann Biomed Eng*. 2007;35(4):546-559. <https://doi.org/10.1007/s10439-006-9240-3>.
37. Vignon-Clementel IE, Figueroa CA, Jansen KE, Taylor CA. Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries. *Comput Methods Appl Mech Eng*. 2006;195(29-32):3776-3796. <https://doi.org/10.1016/j.cma.2005.04.014>.
38. Olufsen M, Nadim A, Lipsitz L. Dynamics of cerebral blood flow regulation explained using a lumped parameter model. *Am J Physiol Regul Integr Comp Physiol*. 2002;282(2):R611-R622. <https://doi.org/10.1152/ajpregu.00285.2001>.
39. Lan H, Updegrave A, Wilson NM, Maher GD, Shadden SC, Marsden AL. A re-engineered software interface and workflow for the open-source Sim Vascular cardiovascular modeling package. *J Biomech Eng*. 2017;140(2):024501. <https://doi.org/10.1115/1.4038751>.

40. Franca LP, Frey SL. Stabilized finite element methods: II. The incompressible Navier-stokes equations. *Comput Methods Appl Mech Eng*. 1992;99(2-3):209-233. [https://doi.org/10.1016/0045-7825\(92\)90041-H](https://doi.org/10.1016/0045-7825(92)90041-H).
41. Ascher UM, Petzold LR. *Computer methods for ordinary differential equations and differential-algebraic equations*. 1st ed. Philadelphia, PA: Society for Industrial and Applied Mathematics; 1998.
42. Saad Y. *Iterative methods for sparse linear systems*. Philadelphia, PA: SIAM; 2003.
43. Wang W, Xu X. A two-level overlapping hybrid domain decomposition method for eigenvalue problems. *SIAM J Numer Anal*. 2018;56(1):344-368. <https://doi.org/10.1137/16M1088302>.
44. Wright GB. Radial basis function interpolation: Numerical and analytical developments. (PhD thesis). Boulder, CO; 2003.
45. Karypis G, Kumar V. Multilevelk-way partitioning scheme for irregular graphs. *J Paral Distrib Comput*. 1998;48(1):96-129. <https://doi.org/10.1006/jpdc.1997.1404>.
46. Balay S, Abhyankar S, Adams MF, et al. PETSc Web page. 2020.
47. Miao Z, Liebeskind DS, Lo W, et al. Fractional flow assessment for the evaluation of intracranial atherosclerosis: a feasibility study. *Interv Neurol*. 2016;5(1-2):65-75. <https://doi.org/10.1159/000444333>.
48. Tonino PA, De Bruyne B, Pijls NH, et al. Fractional flow reserve versus angiography for guiding percutaneous coronary intervention. *New Engl J Med*. 2009;360(3):213-224. <https://doi.org/10.1056/NEJMoa0807611>.
49. Kong F, Kheyfets V, Finol E, Cai XC. An efficient parallel simulation of unsteady blood flows in patient-specific pulmonary artery. *Int J Numer Methods Biomed Eng*. 2018;34(4):e2952. <https://doi.org/10.1002/cnm.2952>.
50. Dongarra J, Foster I, Fox G, et al. *Sourcebook of parallel computing*. San Francisco, CA: Morgan Kaufmann; 2003.

How to cite this article: Chen R, Wu B, Cheng Z, et al. A parallel non-nested two-level domain decomposition method for simulating blood flows in cerebral artery of stroke patient. *Int J Numer Meth Biomed Engng*. 2020;36:e3392. <https://doi.org/10.1002/cnm.3392>