Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 367 (2020) 113122
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

The nested block preconditioning technique for the incompressible Navier–Stokes equations with emphasis on hemodynamic simulations

Ju Liu*, Weiguang Yang, Melody Dong, Alison L. Marsden

Department of Pediatrics (Cardiology), Stanford University, Clark Center E1.3, 318 Campus Drive, Stanford, CA 94305, USA
Department of Bioengineering, Stanford University, Clark Center E1.3, 318 Campus Drive, Stanford, CA 94305, USA
Institute for Computational and Mathematical Engineering, Stanford University, Clark Center E1.3, 318 Campus Drive, Stanford, CA 94305, USA

Received 24 November 2019; received in revised form 25 April 2020; accepted 8 May 2020
Available online 27 May 2020

Abstract

We develop a novel iterative solution method for the incompressible Navier–Stokes equations with boundary conditions coupled with reduced models. The iterative algorithm is designed based on the variational multiscale formulation and the generalized-α scheme. The spatiotemporal discretization leads to a block structure of the resulting consistent tangent matrix in the Newton–Raphson procedure. As a generalization of the conventional block preconditioners, a three-level nested block preconditioner is introduced to attain a better representation of the Schur complement, which plays a key role in the overall algorithm robustness and efficiency. This approach provides a flexible, algorithmic way to handle the Schur complement for problems involving multiscale and multiphysics coupling. The solution method is implemented and benchmarked against experimental data from the nozzle challenge problem issued by the US Food and Drug Administration. The robustness, efficiency, and parallel scalability of the proposed technique are then examined in several settings, including moderately high Reynolds number flows and physiological flows with strong resistance effect due to coupled downstream vasculature models. Two patient-specific hemodynamic simulations, covering systemic and pulmonary flows, are performed to further corroborate the efficacy of the proposed methodology.

Keywords: Variational multiscale method; Saddle-point problem; Nested block preconditioner; Hemodynamics; Geometric multiscale modeling; Patient-specific model

1. Introduction

1.1. Motivation and literature survey

Fast-growing interest in cardiovascular modeling [1,2] and ever-increasing computing power [3] has created a pressing need to efficiently simulate flow physics with high resolution. Running hemodynamic simulations with millions of unknowns has now become a routine part of scientific and clinical research and is now being used routinely in clinical decision-making [4,5]. More and more, there is a growing consensus that proper design of

* Corresponding author.
E-mail addresses: liuju@stanford.edu, liujuy@gmail.com (J. Liu), wgyang@stanford.edu (W. Yang), mldong@stanford.edu (M. Dong), amarsden@stanford.edu (A.L. Marsden).

https://doi.org/10.1016/j.cma.2020.113122
0045-7825/

the preconditioning techniques has become a critical issue for flow simulations, and in particular, hemodynamic simulations. In our opinion, under the setting of parallel computing, a desirable preconditioning technique for flow problems, and especially for hemodynamics, should possess the following attributes.

1. The algorithm should be robust with respect to physical parameters as well as spatiotemporal discretization methodologies.
2. The algorithm should be scalable in terms of fixed-size (strong) scalability as well as isogranular (weak) scalability.
3. The algorithm should require minimum user interventions. Ideally, users should not need to implement new matrices for the definition of the preconditioner.

To achieve the above-mentioned goals, different approaches have been devised to precondition the matrix problem in the setting of the Krylov subspace method. There is a school that favors the idea of domain decomposition (DD) methods because they can be conveniently implemented in the parallel setting with the existing algebraic-based solver reused [6–9]. Although the DD method can be used as a black-box technique with almost no user intervention, its locality nature limits its parallel scalability [10,11], and employing incomplete factorization in subdomains renders it non-robust for saddle-point problems [12]. The recently introduced multi-level DD method borrows concepts from the multigrid method and seems to be a promising direction to overcome the scalability issue [13,14].

The physics-based block preconditioning technique, as an alternative approach, has attracted concentrated research for flow problems [15]. Consider solving a fully implicit scheme for the incompressible Navier–Stokes equations using the consistent Newton–Raphson method, the problem boils down to repeatedly solving a linear system of equations with a 2×2 block structure,

$$\mathcal{A} := \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}.$$

In the above, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ can be regarded as a discrete convection–diffusion–reaction operator, a discrete gradient operator, and a discrete divergence operator, with additional numerical modeling terms, respectively. The matrix $\mathbf{D}$ arises purely due to the subgrid-scale modeling. The matrix $\mathcal{A}$ can be factored into a lower triangular, a diagonal, and an upper triangular matrices as follows,

$$\mathcal{A} = \mathcal{LDU} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{CA}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{O} \\ \mathbf{O} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix},$$

with $\mathbf{S} := \mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}$ being the Schur complement. Therefore, the design of the preconditioner for $\mathcal{A}$ reduces to solving smaller systems associated with $\mathbf{A}$ and $\mathbf{S}$. This technique is attractive because it combines the merits of the Chorin–Teman projection method [16,17] used in the finite volume community and the fully implicit method [18,19] used in the finite element community. Roughly speaking, the physics-based block preconditioner can be regarded as an algebraic procedure that wraps the projection method inside the Krylov iteration. The Schur complement $\mathbf{S}$ can be viewed as an algebraic manifestation of the pressure Poisson equation in the fully implicit scheme. Unlike the classical projection method, the separation of the physical fields does not take place inside the temporal scheme, thus avoiding considerations of artificial pressure boundary conditions and time step size control [20].

A solution method is deemed to be robust with respect to physical parameters if there is no significant impact on its convergence rate with varying physical parameters. For flow problems, this parameter is typically the Reynolds number, which, in hemodynamic simulations, ranges from $\mathcal{O}(10^{-3})$ in the capillary vessels to $\mathcal{O}(10^{3})$ in the aorta. The classical SIMPLE preconditioner extracts the diagonal of $\mathbf{A}$ to define a sparse approximation of the Schur complement. This approach ignores the convection information and is therefore non-robust with respect to the Reynolds number. In fact, it has been a major research thrust to search for a preconditioner that is insensitive to the Reynolds number and at the same time remains scalable in the computational fluid dynamics (CFD) community using fully implicit schemes [10,11,21–27]. Representative examples include the pressure convection–diffusion (PCD) preconditioner [24] and the least-squares commutator (LSC) preconditioner [23]. Both are known to be scalable with respect to discretization resolution and are mildly affected by the Reynolds number. However, they are not without shortcomings. The major drawback of the PCD preconditioner is that it requires the assembly of a new matrix, which is implementationally inconvenient and computationally inefficient. The LSC preconditioner was proposed to remedy that issue by only using the existing blocks in $\mathcal{A}$. However, the LSC preconditioner was

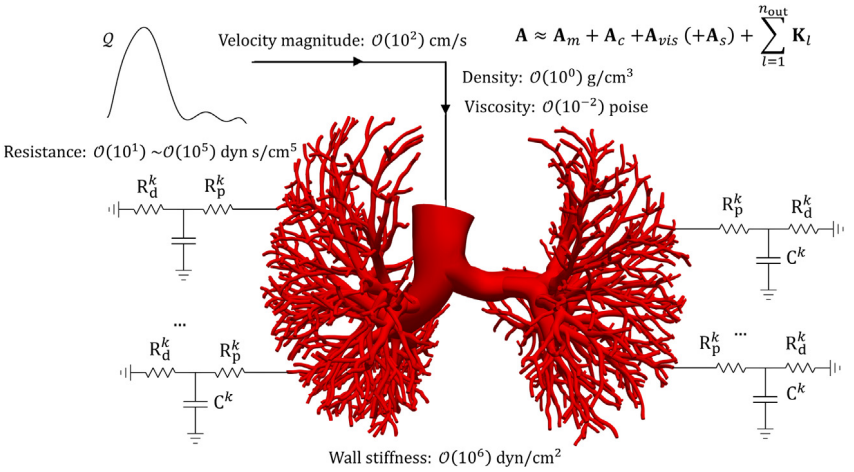


Fig. 1. Illustration of the hemodynamic simulation setting and the structure of the sub-matrix **A**. This sub-matrix can be approximately decomposed into matrices arising from the transient term ($\mathbf{A}_m$), the convection term ($\mathbf{A}_c$), the viscous term ($\mathbf{A}_{vis}$), the reduced models on the outflow boundaries ($\mathbf{K}_l$), and the wall stiffness term if one is considering FSI ($\mathbf{A}_s$). These terms involve physical parameters and the typical magnitude of the physical parameters in cardiovascular simulations is given in the centimeter–gram–second (CGS) units.

proposed and tested based on a mixed formulation using inf–sup stable elements. It has been recently noticed that the LSC preconditioner is quite sensitive to the spatial discretization method, as it cannot converge for the stabilized formulation using equal-order interpolations [11]. The drawbacks of the PCD and LSC preconditioners indicate that the satisfaction of the three attributes listed at the beginning of this section remains a challenging task for the incompressible Navier–Stokes equations.

1.2. A three-level nested block preconditioner

In the setting of blood flow simulations, the situation is more complicated. In addition to the convection term, the downstream vasculature is often modeled as a reduced model [28], which is coupled to the three-dimensional problem via the outflow boundary conditions. This coupling strategy is also referred to as the geometric multiscale modeling [29], and it leads to a modification of the original matrix problem with rank-one matrices multiplied by a factor proportional to the resistance value on the outlet surfaces (see Section 2.5). For realistic problems, the resistance value is not small, and correspondingly the reduced models have a significant impact on the matrix properties. If we look one step further, more challenges arise in the setting of fluid–structure interaction (FSI). Considering the coupled momentum formulation as an example, an additional wall stiffness matrix enters into the tangent matrix [30]. Again, this modification cannot be neglected because the physiologically realistic Young's modulus is by no means small. The contributions to the tangent matrix from different physical sources are illustrated in Fig. 1. *Consequently, over and above the long-lasting efforts for preconditioning the incompressible Navier–Stokes equations, special consideration needs to be further exercised so that the geometric multiscale models and multiphysics coupling, such as FSI, can be properly taken into account in the preconditioner design.*

If one examines the resulting algebraic system carefully, it can be noticed that the rank-one matrices and the wall stiffness matrix enter into the block matrix **A** only, without affecting the other three block matrices. This means that the matrix **A** contains multiple contributions, including the transient term, the viscous term, the convection term, the subgrid scale modeling terms, the rank-one modifications defined on the outlet surfaces, and the stiffness matrix defined on the wall surface if one is solving an FSI problem. The true difficulty comes from designing an effective sparse approximation of the Schur complement that incorporates the information from the above terms. Due to the special algebraic structure of the rank-one matrices, the Sherman–Morrison formula has been utilized to design a Schur complement approximation for hemodynamic simulations [31]. That approach, together with the bi-partitioned iterative algorithm (BIPN) [32], constitutes the backbone solver technology in the current `svSolver` [33], the CFD solver in the SimVascular package [34]. Yet, it is known that the same solver technology performs poorly when

the vessel wall is deformable, and the reason is apparently due to the ignorance of the wall stiffness matrix in the sparse approximation of the Schur complement. Also, one may reasonably expect performance degradation when simulating strong convection problems with `svSolver` since it does not account for the convection terms [31]. As mentioned above, designing a robust and scalable preconditioner for the convection term alone remains an unsettled issue. *Therefore, it will be quite challenging, if not impossible, to explicitly construct an algebraic form of the preconditioning technique accounting for all aforementioned terms simultaneously.* This leads us to alternatively consider constructing the Schur complement approximation via an algorithm, rather than an algebraic form.

Recently, it has been realized that one may apply the so-called Schur complement reduction procedure (SCR) to serve as a preconditioning technique. In the original SCR procedure, one solves the sub-matrices $\mathbf{A}$ and $\mathbf{S}$ to a prescribed tolerance and thereby solves the linear problem associated with $\mathcal{A}$ in one pass [15,35]. Although the excessive memory cost for storing the Schur complement can be resolved by using a matrix-free algorithm to define the action of $\mathbf{S}$ on a vector, solving a linear system associated with $\mathbf{S}$ to a high precision is still prohibitively expensive. To alleviate this issue, one may use the SCR procedure as a preconditioning technique by wrapping it inside an iterative solution method, which leads to a three-level nested algorithm structure (see Section 3.1). In the outer level, the iterative method strives to solve $\mathcal{A}$ either via a static iterative method or a Krylov subspace method; in the intermediate level, the sub-matrices $\mathbf{A}$ and $\mathbf{S}$ are solved (not necessarily to high precision) as a preconditioning technique to accelerate the outer iteration; in the inner level, the matrix-free algorithm for $\mathbf{S}$ necessitates solving $\mathbf{A}$. The inner level solver can be a key ingredient when the matrix $\mathbf{A}$ involves complex contributions from non-traditional sources. Although the introduction of an inner solver may ostensibly increase the computational burden, it can in fact dramatically enhance the solver robustness without losing efficiency if the setting for the inner solver is properly tuned (see Section 4.3). In the conventional physics-based block preconditioner, the sparse approximation of $\mathbf{S}$ is explicitly constructed without invoking the inner level solver, and thence exhibits a two-level structure [36]. In theory, the proposed three-level nested preconditioning technique can be viewed as a generalization of the conventional block preconditioners.

The SCR procedure has been used as a preconditioning technique for CFD problems within the Richardson iteration scheme [37,38] and the biconjugate gradient stabilized method [39]. Those results clearly justified the advantage of using SCR as a preconditioning technique over several standard preconditioners. In our prior work, we investigated the use of FGMRES [40] preconditioned by the SCR procedure for hyperelasticity [41], based on a unified continuum modeling framework [42,43]. New ingredients were added to further enhance the three-level nested algorithm previously introduced in [37–39]. In addition to using the FGMRES algorithm as the outer solver, we applied the GMRES algorithm preconditioned by the algebraic multigrid (AMG) preconditioner at the intermediate and inner levels. This combined the merits of both the conventional block preconditioner [10,11,22] and the nested algorithm proposed in [37–39]. Also, a sparse matrix was constructed to precondition and accelerate the matrix-free solution procedure for $\mathbf{S}$. In this work, we further investigate the efficacy of this solution method for hemodynamic problems, by examining its performance when convection and the geometric multiscale coupling contributions are significant. This study serves as a stepping stone towards a novel iterative solution method for FSI problems, based on our recently proposed FSI framework [42,44] and related iterative solution method for hyperelasticity [41].

1.3. Remarks on the numerical formulation

We want to point out that the numerical formulation adopted in this work is different from those reported in the existing literature in several aspects. First, in several prior works [30,45], an integration-by-parts was performed for the divergence operator in the continuity equation. This approach is not favored because it contradicts with the setting of the pressure function space in the Galerkin formulation [18]. Second, the pressure variable has traditionally been evaluated by the backward Euler method in the generalized-α scheme [46]. This choice will degrade the temporal accuracy, and this issue has recently been rectified by evaluating the pressure at the intermediate time step [47]. Third, in the same vein, the traction forces used to couple the three-dimensional model with the reduced model on the outlet surfaces are evaluated at the intermediate step as well, in contrast to the prior approach [48]. Fourth, the definition of the stabilization parameter in the variational multiscale formulation is modified for simplex elements. This modification makes the stabilization parameter remain invariant under node renumbering and is recently introduced in [49,50].

2. Governing equations and the spatiotemporal discretizations

In this section, we introduce the strong-form problem and the fully discrete problem generated by the variational multiscale formulation and the generalized-α method.

2.1. Strong-form problem

Let $\Omega \subset \mathbb{R}^3$ be a fixed bounded open set with sufficiently smooth (e.g. Lipschitz) boundary $\Gamma := \partial\Omega$. The time interval is denoted $(0, T) \subset \mathbb{R}$ with $T > 0$. The governing equations for the incompressible flow of a Newtonian fluid can be stated as follows.

$$\mathbf{0} = \rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho\boldsymbol{v}\cdot\nabla\boldsymbol{v} - \nabla\cdot\boldsymbol{\sigma} - \rho\boldsymbol{f}, \qquad \text{in } \Omega\times(0,T), \tag{2.1}$$

$$0 = \nabla\cdot\boldsymbol{v}, \qquad \text{in } \Omega\times(0,T), \tag{2.2}$$

wherein

$$\boldsymbol{\sigma} := 2\mu\boldsymbol{\varepsilon}(\boldsymbol{v}) - p\boldsymbol{I}, \quad \boldsymbol{\varepsilon}(\boldsymbol{v}) := \frac{1}{2}\left(\nabla\boldsymbol{v} + \nabla\boldsymbol{v}^T\right). \tag{2.3}$$

In the above, ρ is the fluid density, $\boldsymbol{v}$ is the velocity field, $\boldsymbol{\sigma}$ is the Cauchy stress, $\boldsymbol{f}$ is the body force, μ is the dynamic viscosity, $\boldsymbol{\varepsilon}$ is the rate-of-strain tensor, p is the pressure, and $\boldsymbol{I}$ is the second-order identity tensor. The initial condition is given by a divergence-free velocity field $\boldsymbol{v}_0$ as

$$\boldsymbol{v}(\cdot,0) = \boldsymbol{v}_0(\cdot), \text{ in } \bar{\Omega}. \tag{2.4}$$

The boundary Γ can be partitioned into two non-overlapping subdivisions, that is,

$$\overline{\Gamma} = \overline{\Gamma_g \cup \Gamma_h}, \text{ and } \emptyset = \Gamma_g \cap \Gamma_h. \tag{2.5}$$

In the above, the subscripts g and h indicate the Dirichlet and Neumann partitions, respectively. The unit outward normal vector to Γ is denoted as $\boldsymbol{n}$. Given the Dirichlet data $\boldsymbol{g}$ and the boundary traction $\boldsymbol{h}$, the boundary conditions can be stated as

$$\boldsymbol{v} = \boldsymbol{g} \qquad \text{on } \Gamma_g\times(0,T), \tag{2.6}$$

$$\boldsymbol{\sigma}\boldsymbol{n} = \boldsymbol{h} \qquad \text{on } \Gamma_h\times(0,T). \tag{2.7}$$

2.2. The boundary conditions and geometric multiscale modeling

In the mathematical modeling of cardiovascular biofluids, the description of the boundaries needs to be further refined. We consider that Γ_g can be further partitioned into two non-overlapping subdivisions: $\overline{\Gamma_g} = \overline{\Gamma_{\text{in}} \cup \Gamma_{\text{wall}}}$. On Γ_{in}, the velocity is specified by a prescribed velocity profile $\boldsymbol{v}_{\text{in}}$, while on Γ_{wall}, we impose no-slip boundary condition. This suggests that the Dirichlet data $\boldsymbol{g}$ can be explicitly defined as,

$$\boldsymbol{g} = \begin{cases} \boldsymbol{v}_{\text{in}} & \text{on } \Gamma_{\text{in}}, \\ \mathbf{0} & \text{on } \Gamma_{\text{wall}}. \end{cases}$$

The velocity profiles on the inlet surfaces are typically Poiseuille or Womersley. For patient-specific geometries, the inlet surfaces do not necessarily have a circular shape, and we invoke a special mapping technique introduced in [51] to generate the desired inflow profile on Γ_{in}.

Specialized downstream boundary conditions are essential for capturing the physiologically realistic flow and pressure conditions, and here we use lumped parameter models as the geometric multiscale model to describe the downstream circulation in this work, though other reduced models (e.g. one-dimensional models) can also be employed. Consider the Neumann boundary Γ_h that assumes n_{out} non-overlapping planar surfaces,

$$\Gamma_h = \bigcup_{k=1}^{\text{n}_{\text{out}}} \Gamma_{\text{out}}^k, \qquad \overline{\Gamma_{\text{out}}^i} \cap \overline{\Gamma_{\text{out}}^j} = \emptyset, \text{ for } 1 \le i, j \le \text{n}_{\text{out}}, i \ne j.$$

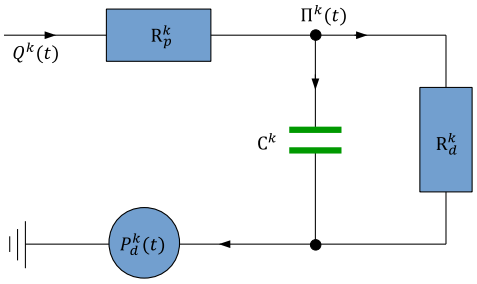


Fig. 2. Schematic representation of the three element Windkessel electric model.

On each outlet surface, the traction boundary condition is defined as

$$\boldsymbol{h}(t) = -P^k(t)\boldsymbol{n} \qquad \text{on } \Gamma^k_{\text{out}},$$

wherein $P^k(\cdot) : \mathbb{R}_+ \to \mathbb{R}$ is a scalar function of time. The flow rate on an outlet surface is defined as

$$Q^k(t) := \int_{\Gamma^k_{\text{out}}} \boldsymbol{v} \cdot \boldsymbol{n} d\Gamma. \tag{2.8}$$

Invoking the Dirichlet-to-Neumann method, the values of P^k at each time instant are determined by solving a set of algebraic-differential equations with the flow rates as inputs. As an example, we consider the following equations known as the three-element Windkessel model,

$$\frac{d\Pi^k(t)}{dt} = \mathcal{F}(\Pi^k(t), Q^k(t)) := -\frac{\Pi^k(t)}{\mathrm{R}^k_{\mathrm{d}}\mathrm{C}^k} + \frac{Q^k(t)}{\mathrm{C}^k}, \tag{2.9}$$

$$P^k(t) = \mathrm{R}^k_{\mathrm{p}} Q^k(t) + \Pi^k(t) + P^k_{\mathrm{d}}(t). \tag{2.10}$$

In the above, C^k, $\mathrm{R}^k_{\mathrm{d}}$, and $\mathrm{R}^k_{\mathrm{p}}$ are constants and represent the compliance, distal resistance, and proximal resistance of the downstream vasculature to each outlet Γ^k_{out}, $\Pi^k(t)$ represents the pressure drop across the distal resistor, and $P^k_{\mathrm{d}}(t)$ represents the distal reference pressure. An electric circuit analogue of this model is illustrated in Fig. 2. For this Windkessel model, one may obtain an analytic representation of $P^k(t)$ in terms of $Q^k(t)$ as follows,

$$P^k(t) = \int_0^t \left(\exp(-\frac{t-s}{\mathrm{R}^k_{\mathrm{d}}\mathrm{C}^k})\frac{Q^k(s)}{\mathrm{C}^k}\right) ds + \mathrm{R}^k_{\mathrm{p}} Q^k(t) + P^k_{\mathrm{d}}(t) + \exp(-\frac{t}{\mathrm{R}^k_{\mathrm{d}}\mathrm{C}^k}) \left(P^k(0) - \mathrm{R}^k_{\mathrm{p}} Q^k(0) - P^k_{\mathrm{d}}(0)\right). \tag{2.11}$$

As a special case, when $\mathrm{C}^k \to 0^+$, the three-element Windkessel model (2.9)–(2.10) reduces to a simpler resistance boundary condition,

$$P^k(t) = \mathrm{R}^k Q^k(t) + P^k_{\mathrm{d}}(t), \tag{2.12}$$

with $\mathrm{R}^k := \mathrm{R}^k_{\mathrm{p}} + \mathrm{R}^k_{\mathrm{d}}$.

Remark 1. In this work, we focus on using the Dirichlet-to-Neumann method to model the physics of the downstream domain. There exists another coupling approach, in which the flow rates are solved via a set of algebraic-differential equations and are passed to the three-dimensional model by imposing velocity profiles on boundary surfaces. This approach, for example, can be used when coupling a heart model with the inlet of an aorta.

2.3. Variational multiscale formulation

The semi-discrete formulation is constructed based on the residual-based variational multiscale method [46]. Let $\mathcal{S}_{\boldsymbol{v}}$ and $\mathcal{S}_p$ denote the trial solution spaces of the fluid velocity and pressure, and let $\mathcal{V}_p$ and $\mathcal{V}_{\boldsymbol{v}}$ be the corresponding test function spaces. These spaces are spanned by finite element basis functions, and in this work, linear polynomials. The Dirichlet boundary condition (2.6) is built into the definition of $\mathcal{S}_{\boldsymbol{v}}$. The union of element interiors is denoted

as Ω'. The semi-discrete formulation can be stated as follows. Find $\boldsymbol{y}_h(t) \coloneqq \{\boldsymbol{v}_h(t), p_h(t)\}^T \in \mathcal{S}_{\boldsymbol{v}} \times \mathcal{S}_p$ such that

$$\mathbf{B}_m\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) = 0, \qquad \forall \boldsymbol{w}_h \in \mathcal{V}_{\boldsymbol{v}}, \tag{2.13}$$

$$\mathbf{B}_p\left(q_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) = 0, \qquad \forall q_h \in \mathcal{V}_p, \tag{2.14}$$

where

$$\mathbf{B}_m\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) \coloneqq \mathbf{B}_m^{\mathrm{vol}}\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) + \mathbf{B}_m^{\mathrm{bc}}\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) + \mathbf{B}_m^{\mathrm{bf}}\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right), \tag{2.15}$$

$$\begin{aligned}\mathbf{B}_m^{\mathrm{vol}}\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) \coloneqq & \int_\Omega \boldsymbol{w}_h \cdot \rho\left(\frac{\partial \boldsymbol{v}_h}{\partial t} + \boldsymbol{v}_h \cdot \nabla \boldsymbol{v}_h - \boldsymbol{b}\right) d\Omega - \int_\Omega \nabla \cdot \boldsymbol{w}_h p_h d\Omega_{\boldsymbol{x}} + \int_\Omega 2\mu \boldsymbol{\varepsilon}(\boldsymbol{w}_h) : \boldsymbol{\varepsilon}(\boldsymbol{v}_h) d\Omega \\ & - \int_{\Omega'} \nabla \boldsymbol{w}_h : \left(\rho \boldsymbol{v}' \otimes \boldsymbol{v}_h\right) d\Omega + \int_{\Omega'} \nabla \boldsymbol{v}_h : \left(\rho \boldsymbol{w}_h \otimes \boldsymbol{v}'\right) d\Omega - \int_{\Omega'} \nabla \boldsymbol{w}_h : \left(\rho \boldsymbol{v}' \otimes \boldsymbol{v}'\right) d\Omega_{\boldsymbol{x}} - \int_{\Omega'} \nabla \cdot \boldsymbol{w}_h p' d\Omega,\end{aligned} \tag{2.16}$$

$$\mathbf{B}_m^{\mathrm{bc}}\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) \coloneqq -\int_{\Gamma_h} \boldsymbol{w}_h \cdot \boldsymbol{h} d\Gamma, \tag{2.17}$$

$$\mathbf{B}_m^{\mathrm{bf}}\left(\boldsymbol{w}_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) \coloneqq -\int_{\Gamma_h} \rho\beta \left(\boldsymbol{v}_h \cdot \boldsymbol{n}\right)_- \boldsymbol{w}_h \cdot \boldsymbol{v}_h d\Gamma, \tag{2.18}$$

$$\mathbf{B}_p\left(q_h; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) \coloneqq \int_\Omega q_h \nabla \cdot \boldsymbol{v}_h d\Omega - \int_{\Omega'} \nabla q_h \cdot \boldsymbol{v}' d\Omega, \tag{2.19}$$

$$\boldsymbol{v}' \coloneqq -\boldsymbol{\tau}_M \left(\rho \frac{\partial \boldsymbol{v}_h}{\partial t} + \rho \boldsymbol{v}_h \cdot \nabla \boldsymbol{v}_h + \nabla p_h - \mu \Delta \boldsymbol{v}^h - \rho \boldsymbol{b}\right), \tag{2.20}$$

$$p' \coloneqq -\tau_C \nabla \cdot \boldsymbol{v}_h, \tag{2.21}$$

$$\boldsymbol{\tau}_M \coloneqq \tau_M \boldsymbol{I}, \tag{2.22}$$

$$\tau_M \coloneqq \frac{1}{\rho}\left(\frac{\mathrm{C_T}}{\Delta t^2} + \boldsymbol{v}_h \cdot \boldsymbol{G} \boldsymbol{v}_h + \mathrm{C_I}\left(\frac{\mu}{\rho}\right)^2 \boldsymbol{G} : \boldsymbol{G}\right)^{-\frac{1}{2}}, \tag{2.23}$$

$$\tau_C \coloneqq \frac{1}{\tau_M \mathrm{tr}\boldsymbol{G}}, \tag{2.24}$$

$$G_{ij} \coloneqq \sum_{k=1}^{3} \frac{\partial \xi_k}{\partial x_i} M_{kl} \frac{\partial \xi_l}{\partial x_j}, \tag{2.25}$$

$$\boldsymbol{M} = [M_{kl}] = \frac{\sqrt[3]{2}}{2}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \tag{2.26}$$

$$\boldsymbol{G} : \boldsymbol{G} \coloneqq \sum_{i,j=1}^{3} G_{ij} G_{ij}, \tag{2.27}$$

$$\mathrm{tr}\boldsymbol{G} \coloneqq \sum_{i=1}^{3} G_{ii}, \tag{2.28}$$

$$\left(\boldsymbol{v}_h \cdot \boldsymbol{n}\right)_- \coloneqq \frac{\boldsymbol{v}_h \cdot \boldsymbol{n} - |\boldsymbol{v}_h \cdot \boldsymbol{n}|}{2} = \begin{cases} \boldsymbol{v}_h \cdot \boldsymbol{n} & \text{if } \boldsymbol{v}_h \cdot \boldsymbol{n} < 0, \\ 0 & \text{if } \boldsymbol{v}_h \cdot \boldsymbol{n} \geq 0. \end{cases} \tag{2.29}$$

In the above, $\boldsymbol{\xi} = \{\xi_i\}_{i=1}^3$ are the coordinates of an element in the parent domain; the value of $\mathrm{C_I}$ relies on the polynomial order of the interpolation basis functions and takes the value of 36 for linear interpolations [52,53]; the value of $\mathrm{C_T}$ is taken to be 4. The term $\mathbf{B}_m^{\mathrm{bf}}$ is an additional term added to enhance the overall numerical robustness in the presence of locally reversed flows near the outlet surfaces [54]. The parameter β is non-dimensional, and its value is fixed to be 0.2 in this work, following the practices in `svSolver` [33].

Remark 2. In (2.26), $\boldsymbol{M}$ is introduced for simplex elements because their standard reference elements are not symmetric, and nodal permutations may lead to changes in the definition $\boldsymbol{G}$ without using $\boldsymbol{M}$. The entries of $\boldsymbol{M}$

are obtained by mapping the reference element to a regular simplex without changing the element volume. For a detailed derivation of $\boldsymbol{M}$ and its variants in two- and four-dimensions, readers are referred to [49,50].

Remark 3. The term (2.18) is introduced as a modification of the Neumann boundary condition to compensate for the incoming energy due to locally reversed flows. It can be shown that choosing $\beta = 1.0$ leads to an energy stable formulation [55]. In practice, it is found numerically that choosing β smaller than 1.0 is often sufficient for robust performances [56] and is apparently less intrusive for flow physics. For a recent summary of the treatment for backflow instabilities, readers are referred to [57].

2.4. Temporal discretization

To derive the fully discrete formulation for problems with geometric multiscale coupled boundary conditions, we apply two time-stepping schemes for the three-dimensional and zero-dimensional domains separately. In the three-dimensional domain, the generalized-α method is applied; in the zero-dimensional domain, an explicit fourth-order Runge–Kutta method is utilized with subdivided time intervals [48].

2.4.1. Temporal discretization of the three-dimensional problem

Let the time interval $(0, T)$ be divided into a set of N_{ts} subintervals of size $\Delta t_n := t_{n+1} - t_n$, which is delimited by a discrete time vector $\{t_n\}_{n=0}^{N_{\text{ts}}}$. The approximations to the solution vector and its first time derivative evaluated at the time step t_n are denoted as $\boldsymbol{y}_n := \{\boldsymbol{v}_n, p_n\}^T$ and $\dot{\boldsymbol{y}}_n := \{\dot{\boldsymbol{v}}_n, \dot{p}_n\}^T$. The approximation to $\boldsymbol{h}(t)$ at time t_n is denoted as $\boldsymbol{h}_n$. Let $\boldsymbol{e}_i$ be the Cartesian basis vector with $i = 1, 2, 3$ and N_A be the basis function of the discrete function spaces for the velocity component as well as pressure. We may then define the residual vectors as

$$\mathbf{R}_m^{\text{vol}}\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) := \left\{\mathbf{B}_m^{\text{vol}}\left(N_A \boldsymbol{e}_i; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\}, \qquad \mathbf{R}_m^{\text{bc}}\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) := \left\{\mathbf{B}_m^{\text{bc}}\left(N_A \boldsymbol{e}_i; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\},$$
$$\mathbf{R}_m^{\text{bf}}\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) := \left\{\mathbf{B}_m^{\text{bf}}\left(N_A \boldsymbol{e}_i; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\}, \qquad \mathbf{R}_m\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) := \left\{\mathbf{B}_m\left(N_A \boldsymbol{e}_i; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\},$$
$$\mathbf{R}_p\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) := \left\{\mathbf{B}_p\left(N_A; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\}.$$

With the above notations, we have $\mathbf{R}_m\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) = \mathbf{R}_m^{\text{vol}}\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) + \mathbf{R}_m^{\text{bc}}\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) + \mathbf{R}_m^{\text{bf}}\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)$ due to their definitions. The fully discrete scheme can be stated as follows. At time step t_n, given $\dot{\boldsymbol{y}}_n$, $\boldsymbol{y}_n$, and the time step size Δt_n, find $\dot{\boldsymbol{y}}_{n+1}$ and $\boldsymbol{y}_{n+1}$ such that

$$\mathbf{R}_m(\dot{\boldsymbol{y}}_{n+\alpha_m}, \boldsymbol{y}_{n+\alpha_f}) = \mathbf{0}, \tag{2.30}$$
$$\mathbf{R}_p(\dot{\boldsymbol{y}}_{n+\alpha_m}, \boldsymbol{y}_{n+\alpha_f}) = \mathbf{0}, \tag{2.31}$$
$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \dot{\boldsymbol{y}}_n, + \gamma \Delta t_n \left(\dot{\boldsymbol{y}}_{n+1} - \dot{\boldsymbol{y}}_n\right), \tag{2.32}$$
$$\dot{\boldsymbol{y}}_{n+\alpha_m} = \dot{\boldsymbol{y}}_n + \alpha_m \left(\dot{\boldsymbol{y}}_{n+1} - \dot{\boldsymbol{y}}_n\right), \tag{2.33}$$
$$\boldsymbol{y}_{n+\alpha_f} = \boldsymbol{y}_n + \alpha_f \left(\boldsymbol{y}_{n+1} - \boldsymbol{y}_n\right). \tag{2.34}$$

In the above system, there are three parameters α_m, α_f and γ, whose values determine critical numerical properties of the discrete dynamic system. To ensure second-order accuracy and unconditional stability (for linear problems), and optimal high frequency dissipation, they are parametrized as

$$\alpha_m = \frac{1}{2}\left(\frac{3 - \varrho_\infty}{1 + \varrho_\infty}\right), \quad \alpha_f = \frac{1}{1 + \varrho_\infty}, \quad \gamma = \frac{1}{1 + \varrho_\infty},$$

wherein $\varrho_\infty \in [0, 1]$ denotes the spectral radius of the amplification matrix at the highest mode [19,58]. Except the simulations presented in Section 4.1, we choose $\varrho_\infty = 0.5$ in the generalized-α method.

Remark 4. In the literature, the pressure is often collocated at the time step $n + 1$ rather than $n + \alpha_f$ within the generalized-α scheme [30,45,46,48]. In a recent investigation, we found that the widely-adopted choice leads to a first-order temporal accuracy, at least, for the pressure [47]. Evaluating the pressure at the intermediate time step $n + \alpha_f$ not only recovers the second order accuracy but also simplifies the implementation.

2.4.2. Temporal discretization of the zero-dimensional problem

Considering the implicitly coupled outflow boundary condition given by the Windkessel model (2.30), one needs to evaluate the values of the proximal pressure at time $t_{n+\alpha_f}$. Although an analytic solution (2.11) exists for the specific model considered in this work, it is typically unavailable for general lumped parameter models. Therefore, we consider performing a time integration for the algebraic-differential equations (2.9)–(2.10) for its evaluation. Following the framework proposed in [48], we solve the ordinary differential equations (2.9) via an explicit fourth-order Runge–Kutta method by subdividing the time interval (t_n, t_{n+1}) into n_{ts} equal-sized subintervals $(t_{n,m}, t_{n,m+1})$ that satisfies

$$t_{n,m} := t_n + mh, \qquad \text{for } m = 0, \dots, n_{\text{ts}}, \ \text{with } h := \frac{\Delta t_n}{n_{\text{ts}}}.$$

We denote by $\Pi^k_{n,m}$ the approximations to $\Pi^k(t_{n,m})$, and we may define the approximation to the flow rate at the intermediate time step $t_{n,m}$ by linear interpolations,

$$Q^k_{n,m} := \left(1 - \frac{m}{n_{\text{ts}}}\right) Q^k_{n,0} + \frac{m}{n_{\text{ts}}} Q^k_{n,n_{\text{ts}}} = \left(1 - \frac{m}{n_{\text{ts}}}\right) Q^k_n + \frac{m}{n_{\text{ts}}} Q^k_{n+1}, \qquad \text{for } m = 0, \dots, n_{\text{ts}}.$$

In the above, the values of Q^k_n and Q^k_{n+1} are explicitly calculated by

$$Q^k_{n,0} = Q^k_n := \int_{\Gamma^k_{\text{out}}} \boldsymbol{v}_n \cdot \boldsymbol{n} d\Gamma, \quad Q^k_{n,n_{\text{ts}}} = Q^k_{n+1} := \int_{\Gamma^k_{\text{out}}} \boldsymbol{v}_{n+1} \cdot \boldsymbol{n} d\Gamma.$$

Given the values of Π^k_n, Q^k_n, and Q^k_{n+1} as the input data, the algorithm for obtaining P^k_{n+1}, the approximation to $P^k(t_{n+1})$, is stated in Algorithm 1. To simplify notations, we denote the dependency of P^k_{n+1} on the input data Π^k_n, Q^k_n, and Q^k_{n+1} through Algorithm 1 as $P^k_{n+1} = \mathcal{G}(\Pi^k_n, Q^k_n, Q^k_{n+1})$ in the following text.

Algorithm 1 The fourth-order Runge–Kutta method for solving Eqs. (2.9)–(2.10).

Set $\Pi^k_{n,0} \leftarrow \Pi^k_n$, $Q^k_{n,0} \leftarrow Q^k_n$, and $Q^k_{n,n_{\text{ts}}} \leftarrow Q^k_{n+1}$
for $m = 0$ to $n_{\text{ts}} - 1$ **do**
  Calculate $K_1 \leftarrow \mathcal{F}\left(\Pi^k_{n,m}, Q^k_{n,m}\right)$
  Calculate $K_2 \leftarrow \mathcal{F}\left(\Pi^k_{n,m} + \frac{1}{3}K_1 h, \frac{2}{3}Q^k_{n,m} + \frac{1}{3}Q^k_{n,m+1}\right)$
  Calculate $K_3 \leftarrow \mathcal{F}\left(\Pi^k_{n,m} - \frac{1}{3}K_1 h + K_2 h, \frac{1}{3}Q^k_{n,m} + \frac{2}{3}Q^k_{n,m+1}\right)$
  Calculate $K_4 \leftarrow \mathcal{F}\left(\Pi^k_{n,m} + K_1 h - K_2 h + K_3 h, Q^k_{n,m+1}\right)$
  Calculate $\Pi^k_{n,m+1} \leftarrow \Pi^k_{n,m} + \frac{1}{8}K_1 h + \frac{3}{8}K_2 h + \frac{3}{8}K_3 h + \frac{1}{8}K_4 h$
end for
return $P^k_{n+1} \leftarrow \mathrm{R}^k_{\text{p}} Q^k_{n+1} + \Pi^k_{n,n_{\text{ts}}} + P^k_d(t_{n+1})$

2.5. Predictor multi-corrector algorithm

The system of Eqs. (2.30)–(2.34) are solved using the Newton–Raphson method with consistent linearization. At the time step t_{n+1}, the solution vector $\boldsymbol{y}_{n+1}$ is solved by the following predictor multi-corrector algorithm. We denote $\boldsymbol{y}_{n+1,(l)} := \left\{\boldsymbol{v}_{n+1,(l)}, p_{n+1,(l)}\right\}^T$ as the solution vector for the three-dimensional problem evaluated at the Newton–Raphson iteration step $l = 0, \dots, l_{\max}$. The residual vectors evaluated at the iteration stage l are denoted as

$$\mathbf{R}_{(l)} := \left\{\mathbf{R}_{m,(l)}, \mathbf{R}_{p,(l)}\right\}^T,$$

with

$$\mathbf{R}^{\text{vol}}_{m,(l)} := \mathbf{R}^{\text{vol}}_m\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right), \qquad \mathbf{R}^{\text{bc}}_{m,(l)} := \mathbf{R}^{\text{bc}}_m\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right),$$

$$\mathbf{R}^{\text{bf}}_{m,(l)} := \mathbf{R}^{\text{bf}}_m\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right), \qquad \mathbf{R}_{m,(l)} := \mathbf{R}_m\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right) = \mathbf{R}^{\text{vol}}_{m,(l)} + \mathbf{R}^{\text{bc}}_{m,(l)} + \mathbf{R}^{\text{bf}}_{m,(l)},$$

$$\mathbf{R}_{p,(l)} := \mathbf{R}_p\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right).$$

In the above, the term $\mathbf{R}^{\text{bc}}_{m,(l)}$ can be explicitly written as

$$\mathbf{R}^{\text{bc}}_{m,(l)} = \sum_{k=1}^{\text{n}_{\text{out}}} \int_{\Gamma^k_{\text{out}}} P^k_{n+\alpha_f,(l)} N_A n_i d\Gamma = \sum_{k=1}^{\text{n}_{\text{out}}} \int_{\Gamma^k_{\text{out}}} \left((1-\alpha_f) P^k_n + \alpha_f P^k_{n+1,(l)}\right) N_A n_i d\Gamma.$$

With $\boldsymbol{v}_{n+1,(l)}$, the corresponding flow rate at the Γ^k_{out} surface is denoted by $Q^k_{n+1,(l)}$. We use $P^k_{n+1,(l)}$ to represent the solution of the zero-dimensional problem solved with Π^k_n, Q^k_n, and $Q^k_{n+1,(l)}$ as the input data for Algorithm 1. The consistent tangent matrix associated with the above residual vectors is

$$\boldsymbol{\mathcal{A}}_{(l)} = \begin{bmatrix} \mathbf{A}_{(l)} & \mathbf{B}_{(l)} \\ \mathbf{C}_{(l)} & \mathbf{D}_{(l)} \end{bmatrix}, \tag{2.35}$$

wherein

$$\mathbf{A}_{(l)} = \mathbf{F}_{(l)} + \mathbf{K}_{(l)}, \tag{2.36}$$

$$\mathbf{F}_{(l)} = \alpha_m \frac{\partial \mathbf{R}^{\text{vol}}_m \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \dot{\boldsymbol{v}}_{n+\alpha_m,(l)}} + \alpha_f \gamma \Delta t_n \frac{\partial \mathbf{R}^{\text{vol}}_m \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{v}_{n+\alpha_f,(l)}} \tag{2.37}$$

$$+ \alpha_f \gamma \Delta t_n \frac{\partial \mathbf{R}^{\text{bf}}_m \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{v}_{n+\alpha_f,(l)}}, \tag{2.38}$$

$$\mathbf{K}_{(l)} = \alpha_f \gamma \Delta t_n \frac{\partial \mathbf{R}^{\text{bc}}_m \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{v}_{n+\alpha_f,(l)}}, \tag{2.39}$$

$$\mathbf{B}_{(l)} = \alpha_f \gamma \Delta t_n \frac{\partial \mathbf{R}^{\text{vol}}_m \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{p}_{n+\alpha_f,(l)}}, \tag{2.40}$$

$$\mathbf{C}_{(l)} = \alpha_m \frac{\partial \mathbf{R}_p \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \dot{\boldsymbol{v}}_{n+\alpha_m,(l)}} + \alpha_f \gamma \Delta t_n \frac{\partial \mathbf{R}_p \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{v}_{n+\alpha_f,(l)}}, \tag{2.41}$$

$$\mathbf{D}_{(l)} = \alpha_f \gamma \Delta t_n \frac{\partial \mathbf{R}_p \left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{p}_{n+\alpha_f,(l)}}. \tag{2.42}$$

Specifically, $\mathbf{K}_{(l)}$ can be explicitly expressed as

$$\mathbf{K}_{(l)} = \alpha_f \gamma \Delta t_n \sum_{k=1}^{\text{n}_{\text{out}}} \left(\mathfrak{m}^k_{(l)} \boldsymbol{a}^k \boldsymbol{a}^{kT}\right), \tag{2.43}$$

wherein $\boldsymbol{a}^k$ is a column vector with its entries defined as

$$\boldsymbol{a}^k_{Ai} := \left[\int_{\Gamma^k_{\text{out}}} N_A \boldsymbol{n}^k_i d\Gamma\right],$$

and $\mathfrak{m}^k_{(l)}$ is a scalar variable defined as

$$\mathfrak{m}^k_{(l)} := \frac{\partial P^k_{n+\alpha_f,(l)}}{\partial Q^k_{n+\alpha_f,(l)}} = \frac{\partial P^k_{n+\alpha_f,(l)}}{\partial P^k_{n+1,(l)}} \frac{\partial P^k_{n+1,(l)}}{\partial Q^k_{n+1,(l)}} \frac{\partial Q^k_{n+1,(l)}}{\partial Q^k_{n+\alpha_f,(l)}} = \alpha_f \frac{\partial P^k_{n+1,(l)}}{\partial Q^k_{n+1,(l)}} \frac{1}{\alpha_f} = \frac{\partial P^k_{n+1,(l)}}{\partial Q^k_{n+1,(l)}}.$$

The matrix $\mathbf{K}_{(l)}$ is a weighted sum of rank-one matrices $\boldsymbol{a}^k \boldsymbol{a}^{kT}$, with weights being $\alpha_f \gamma \Delta t_n \mathfrak{m}^k_{(l)}$. When the outflow boundaries are prescribed by the resistance boundary condition (2.12), we directly have $\mathfrak{m}^k_{(l)} = \text{R}^k$. Yet, for general reduced models, the values of $\mathfrak{m}^k_{(l)}$ can only be obtained through a difference approximation. One first obtains

$$\tilde{P}^k_{n+1,(l)} := \mathcal{G}(\Pi^k_n, Q^k_n, Q^k_{n+1,(l)} + \epsilon/2) \quad \text{and} \quad \hat{P}^k_{n+1,(l)} := \mathcal{G}(\Pi^k_n, Q^k_n, Q^k_{n+1,(l)} - \epsilon/2)$$

by calling Algorithm 1 and then calculates

$$m_{(l)}^k \approx \frac{\tilde{P}_{n+1,(l)}^k - \hat{P}_{n+1,(l)}^k}{\epsilon}, \text{ with } \epsilon := \max\left\{\epsilon_{\text{abs}}, \epsilon_{\text{rel}}|Q_{n+1,(l)}^k|\right\}. \tag{2.44}$$

Following the choice made in the `svSolver`, we fix $\epsilon_{\text{abs}} = 10^{-8}$ and $\epsilon_{\text{rel}} = 10^{-5}$. Based on the above discussion, the predictor multi-corrector algorithm for solving the nonlinear algebraic equations in each time step can be summarized as follows.

Predictor stage: Set:

$$\boldsymbol{y}_{n+1,(0)} = \boldsymbol{y}_n, \quad \dot{\boldsymbol{y}}_{n+1,(0)} = \frac{\gamma - 1}{\gamma}\dot{\boldsymbol{y}}_n.$$

Multi-corrector stage: Repeat the following steps for $l = 1, \ldots, l_{\max}$:

1. Calculate the flow rate
$$Q_{n+1,(l)}^k := \int_{\Gamma_{\text{out}}^k} \boldsymbol{v}_{n+1,(l)} \cdot \boldsymbol{n} d\Gamma.$$
2. Calculate
$$P_{n+1,(l)}^k = \mathcal{G}(\Pi_n^k, Q_n^k, Q_{n+1,(l)}^k),$$
$$\tilde{P}_{n+1,(l)}^k = \mathcal{G}(\Pi_n^k, Q_n^k, Q_{n+1,(l)}^k + \frac{1}{2}\epsilon),$$
$$\hat{P}_{n+1,(l)}^k = \mathcal{G}(\Pi_n^k, Q_n^k, Q_{n+1,(l)}^k - \frac{1}{2}\epsilon),$$
by invoking Algorithm 1.
3. Calculate $m_{(l)}^k$ from the difference quotient (2.44).
4. Evaluate the solution vectors at the intermediate stages:
$$\boldsymbol{y}_{n+\alpha_f,(l)} = \boldsymbol{y}_n + \alpha_f\left(\boldsymbol{y}_{n+1,(l-1)} - \boldsymbol{y}_n\right), \quad \dot{\boldsymbol{y}}_{n+\alpha_m,(l)} = \dot{\boldsymbol{y}}_n + \alpha_m\left(\dot{\boldsymbol{y}}_{n+1,(l-1)} - \dot{\boldsymbol{y}}_n\right).$$
5. Evaluate $P_{n+\alpha_f,(l)}^k = (1-\alpha_f)P_n^k + \alpha_f P_{n+1,(l)}^k$.
6. Assemble the residual vectors $\mathbf{R}_{m,(l)}$ and $\mathbf{R}_{p,(l)}$ using $\boldsymbol{y}_{n+\alpha_f,(l)}$, $\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}$, and $P_{n+\alpha_f,(l)}^k$.
7. Let $\|\mathbf{R}_{(l)}\|_{l^2}$ denote the l^2-norm of the residual vector. If either one of the following stopping criteria
$$\frac{\|\mathbf{R}_{(l)}\|_{l^2}}{\|\mathbf{R}_{(0)}\|_{l^2}} \le \text{tol}_{\text{R}}, \qquad \|\mathbf{R}_{(l)}\|_{l^2} \le \text{tol}_{\text{A}},$$
is satisfied for two prescribed tolerances tol_{R}, tol_{A}, set the solution vector at time step t_{n+1} as $\dot{\boldsymbol{y}}_{n+1} = \dot{\boldsymbol{y}}_{n+1,(l-1)}$ and $\boldsymbol{y}_{n+1} = \boldsymbol{y}_{n+1,(l-1)}$, and exit the multi-corrector stage; otherwise, continue to step 8.
8. Assemble the tangent matrices (2.35)–(2.42).
9. Solve the following linear system of equations for $\Delta\dot{p}_{n+1,(l)}$ and $\Delta\dot{\boldsymbol{v}}_{n+1,(l)}$,
$$\begin{bmatrix}\mathbf{A}_{(l)} & \mathbf{B}_{(l)} \\ \mathbf{C}_{(l)} & \mathbf{D}_{(l)}\end{bmatrix}\begin{bmatrix}\Delta\dot{\boldsymbol{v}}_{n+1,(l)} \\ \Delta\dot{p}_{n+1,(l)}\end{bmatrix} = -\begin{bmatrix}\mathbf{R}_{m,(l)} \\ \mathbf{R}_{p,(l)}\end{bmatrix}. \tag{2.45}$$
10. Let $\Delta\dot{\boldsymbol{y}}_{n+1,(l)} := \left\{\Delta\dot{\boldsymbol{v}}_{n+1,(l)}, \Delta\dot{p}_{n+1,(l)}\right\}^T$. Update the solution vectors as
$$\boldsymbol{y}_{n+1,(l)} = \boldsymbol{y}_{n+1,(l)} + \gamma\Delta t_n \Delta\dot{\boldsymbol{y}}_{n+1,(l)}, \quad \dot{\boldsymbol{y}}_{n+1,(l)} = \dot{\boldsymbol{y}}_{n+1,(l)} + \Delta\dot{\boldsymbol{y}}_{n+1,(l)},$$
and return to step 1.

For the numerical simulations presented in this work, unless otherwise specified, we adopt the tolerances for the nonlinear iteration as $\text{tol}_{\text{R}} = \text{tol}_{\text{A}} = 10^{-6}$ and the maximum number of iterations as $l_{\max} = 20$. Notice that in step 5, we evaluated the pressure from the reduced model at the intermediate time step $n + \alpha_f$ to make it consistent with the generalized-α scheme given in Section 2.4.1. This is different from the approach adopted in prior studies [48], in which the pressure from the reduced model and the pressure in the three-dimensional problem are both collocated at the time step $n + 1$.

Remark 5. In the literature, the modified Newton–Raphson method is often used by neglecting some terms in $\mathcal{A}_{(l)}$ [46, p. 183]. This gives a better conditioned tangent matrix while sacrificing the quadratic convergence characteristic of the Newton–Raphson method. In this work, the consistent Newton–Raphson method is utilized.

3. Iterative solution method with the nested block preconditioning

In this section, we focus on developing the iterative solution method for (2.45), which is the most time consuming part in the predictor multi-corrector algorithm. For notational simplicity, we neglect the subscript (l) used to represent the iteration steps in the multi-corrector stage. In this section, we consider the linear system $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$, where the matrix and vectors adopt the following block structure,

$$\mathcal{A} \coloneqq \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \quad \boldsymbol{x} \coloneqq \begin{bmatrix} \boldsymbol{x}_v \\ \boldsymbol{x}_p \end{bmatrix}, \quad \boldsymbol{r} \coloneqq \begin{bmatrix} \boldsymbol{r}_v \\ \boldsymbol{r}_p \end{bmatrix},$$

with

$$\mathbf{A} = \mathbf{F} + \mathbf{K}, \quad \text{and} \quad \mathbf{K} = \alpha_f \gamma \Delta t_n \sum_{k=1}^{\mathrm{n_{out}}} \left(\mathfrak{m}^k \boldsymbol{a}^k \boldsymbol{a}^{kT} \right).$$

For a sparse non-symmetric matrix like $\mathcal{A}$, the GMRES algorithm and its variants [40,59], with a suitable preconditioning technique, are among the most effective general-purpose solution method. In this work, we consider using the FGMRES [40] as the iterative solution method, since the proposed preconditioner varies over iterations. Let $\mathcal{P}_{(i)}$ denote the preconditioning matrix in the ith iteration. In the FGMRES algorithm with right preconditioning, one strives to generate a subspace by repeatedly applying $\mathcal{A}\mathcal{P}_{(i)}^{-1}$ to a vector, and search for an approximate solution by minimizing the residual over the subspace. This procedure is referred to as the *outer solver* in this work. Similar to the GMRES algorithm, the FGMRES algorithm is restarted every m steps and is denoted as FGMRES(m). It is worth mentioning that there are also caveats associated with the FGMRES algorithm. It requires slightly more memory to store the generated subspace. More critically, there is no general convergence theory for it, since it does not generate a standard Krylov subspace. In practice, one might observe stagnation or even divergence due to an improper choice of the preconditioner. Therefore, a suitable design of $\mathcal{P}_{(i)}$ has become the crux. With the understanding that the preconditioners may vary over the iterations, we neglect the subscript (i) for notational simplicity. In Section 3.1, we focus on designing a preconditioning technique based on the SCR procedure, denoted as $\mathcal{P}_{\mathrm{SCR}}$. Following that, we will review the SIMPLE preconditioner $\mathcal{P}_{\mathrm{SIMPLE}}$ in Section 3.2 and the iterative algorithm used in the current `svSolver` [33] in Section 3.3.

3.1. Schur complement reduction

In this section, we introduce the concept of SCR and design a preconditioning technique based on it. The matrix $\mathcal{A}$ can be factorized as follows,

$$\mathcal{A} = \mathcal{L}\mathcal{D}\mathcal{U} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{O} \\ \mathbf{O} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}, \tag{3.1}$$

with $\mathbf{S} \coloneqq \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ being the Schur complement of $\mathbf{A}$. This implies that we may solve $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$ by considering

$$\mathcal{D}\mathcal{U}\boldsymbol{x} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{O} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_v \\ \boldsymbol{x}_p \end{bmatrix} = \mathcal{L}^{-1}\boldsymbol{r} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{r}_v \\ \boldsymbol{r}_p \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_v \\ \boldsymbol{r}_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_v \\ \boldsymbol{r}_p - \mathbf{C}\mathbf{A}^{-1}\boldsymbol{r}_v \end{bmatrix}.$$

The above system of equations can then be solved by a back substitution, which results in the SCR procedure [15,35]. Therefore, the design of the solution method for the matrix problem $\mathcal{A}$ reduces to the design of solution methods for matrices $\mathbf{A}$ and $\mathbf{S}$. Since the algebraic form of the matrix $\mathbf{A}$ is available, one may apply GMRES($\mathrm{m_A}$) with a suitable preconditioner $\mathbf{P}_{\mathrm{A}}$ to solve it. In this work, we consider two options for the preconditioner. We use AMG for robust and scalable performances, and we use the Jacobi preconditioner to achieve efficient performances. The stopping criteria for solving $\mathbf{A}$ include the tolerance for the relative error δ_{A}^r, the tolerance for the absolute error δ_{A}^a, and the maximum number of iterations $\mathrm{n}_{\mathrm{A}}^{\max}$.

The difficulty of the SCR procedure comes from solving the equations associated with $\mathbf{S}$ because the Schur complement $\mathbf{S}$ is defined as a composition of all four block matrices. The appearance of $\mathbf{A}^{-1}$ inevitably makes $\mathbf{S}$

a dense matrix. Therefore, forming $\mathbf{S}$ algebraically is impractical primarily due to the limit of computer memory sizes. Recall that for all iterative methods, one just needs to repeatedly perform matrix–vector multiplications. Therefore, the application of $\mathbf{S}$ on a vector can be achieved without an algebraic definition of it, and this procedure is summarized as the matrix-free algorithm in Algorithm 2. Notice that in this algorithm one has to solve an equation associated with $\mathbf{A}$. This procedure is referred to as the *inner solver*, as it resides inside the solution procedure for $\mathbf{S}$. In practice, we invoke the same iterative method and the same preconditioning technique for equations associated with $\mathbf{A}$ in the inner solver with a potentially different tolerance for the relative error δ_{I}^r. One may call a GMRES($\mathrm{m_S}$) algorithm to solve $\mathbf{S}$ using Algorithm 2 to define the action of the matrix on a vector. We may further accelerate the algorithm by providing a preconditioner for this GMRES iteration. A sparse matrix $\hat{\mathbf{S}} \coloneqq \mathbf{D} - \mathbf{C}\left(\mathrm{diag}\left(\mathbf{A}\right)\right)^{-1}\mathbf{B}$ can be constructed as an approximation to $\mathbf{S}$, and then an AMG preconditioner based on $\hat{\mathbf{S}}$ can be constructed and applied as a left preconditioner in the GMRES iterations. This preconditioner is denoted by $\mathbf{P}_{\mathrm{S}}$. The stopping criteria for solving $\mathbf{S}$ include the tolerance for the relative error δ_{S}^r, the tolerance for the absolute error δ_{S}^a, and the maximum number of iterations $\mathrm{n_S^{max}}$.

Algorithm 2 The matrix-free algorithm for the multiplication of $\mathbf{S}$ with a vector $\boldsymbol{x}_p$.

1: Compute the matrix-vector multiplication $\hat{\boldsymbol{x}}_p \leftarrow \mathbf{D}\boldsymbol{x}_p$.
2: Compute the matrix-vector multiplication $\bar{\boldsymbol{x}}_p \leftarrow \mathbf{B}\boldsymbol{x}_p$.
3: Solve for $\tilde{\boldsymbol{x}}_p$ from the linear system

$$\mathbf{A}\tilde{\boldsymbol{x}}_p = \bar{\boldsymbol{x}}_p \tag{3.2}$$

by GMRES($\mathrm{m_A}$) preconditioned by $\mathbf{P}_{\mathrm{A}}$ with δ_{I}^r, δ_{A}^a, and $\mathrm{n_A^{max}}$ prescribed.
4: Compute the matrix--vector multiplication $\bar{\boldsymbol{x}}_p \leftarrow \mathbf{C}\tilde{\boldsymbol{x}}_p$. ▷ The vector $\bar{\boldsymbol{x}}_p$ is reused.
5: **return** $\hat{\boldsymbol{x}}_p - \bar{\boldsymbol{x}}_p$.

With the solution method for the matrices $\mathbf{A}$ and $\mathbf{S}$ defined, we may define the preconditioner $\mathcal{P}_{\mathrm{SCR}}$. Given the solution method GMRES($\mathrm{m_A}$), GMRES($\mathrm{m_S}$), the preconditioner $\mathbf{P}_{\mathrm{A}}$, $\mathbf{P}_{\mathrm{S}}$, the relative tolerances δ_{A}^r, δ_{S}^r, and δ_{I}^r, the absolute tolerances δ_{A}^a, δ_{S}^a, and the maximum number of iterations $\mathrm{n_A^{max}}$, $\mathrm{n_S^{max}}$, the action of $\mathcal{P}_{\mathrm{SCR}}^{-1}$ on a vector can be summarized as Algorithm 3 in below.

Algorithm 3 The action of $\mathcal{P}_{\mathrm{SCR}}^{-1}$ on a vector $\boldsymbol{s} \coloneqq [\boldsymbol{s}_v; \boldsymbol{s}_p]^T$ with the output being $\boldsymbol{y} \coloneqq [\boldsymbol{y}_v; \boldsymbol{y}_p]^T$.

1: Solve for an intermediate velocity $\hat{\boldsymbol{y}}_v$ from the equation

$$\mathbf{A}\hat{\boldsymbol{y}}_v = \boldsymbol{s}_v \tag{3.3}$$

by GMRES($\mathrm{m_A}$) preconditioned by $\mathbf{P}_{\mathrm{A}}$ with δ_{A}^r, δ_{A}^a, and $\mathrm{n_A^{max}}$ prescribed.
2: Update the continuity residual by $\boldsymbol{s}_p \leftarrow \boldsymbol{s}_p - \mathbf{C}\hat{\boldsymbol{y}}_v$.
3: Solve for $\boldsymbol{y}_p$ from the equation

$$\mathbf{S}\boldsymbol{y}_p = \boldsymbol{s}_p \tag{3.4}$$

by Algorithm 2 and GMRES($\mathrm{m_S}$) preconditioned by $\mathbf{P}_{\mathrm{S}}$ with δ_{S}^r, δ_{S}^a, and $\mathrm{n_S^{max}}$ prescribed.
4: Update the momentum residual by $\boldsymbol{s}_v \leftarrow \boldsymbol{s}_v - \mathbf{B}\boldsymbol{y}_p$.
5: Solve for $\boldsymbol{y}_v$ from the equation

$$\mathbf{A}\boldsymbol{y}_v = \boldsymbol{s}_v \tag{3.5}$$

by GMRES($\mathrm{m_A}$) preconditioned by $\mathbf{P}_{\mathrm{A}}$ with δ_{A}^r, δ_{A}^a, and $\mathrm{n_A^{max}}$ prescribed.

The three solution procedures for (3.3)–(3.5) are referred to as *intermediate solvers*. The preconditioner $\mathcal{P}_{\mathrm{SCR}}$ is implicitly defined through the intermediate solvers, and therefore it varies over iterations. In this work, the absolute tolerances are set to be $\delta_{\mathrm{A}}^a = \delta_{\mathrm{S}}^a = 10^{-50}$. When the preconditioners $\mathbf{P}_{\mathrm{A}}$ and $\mathbf{P}_{\mathrm{S}}$ are constructed by AMG, we adopt the implementation of the BoomerAMG preconditioner from the Hypre package, with its settings summarized in Table 1. Unless otherwise specified, we choose $\mathrm{m_A} = \mathrm{m_S} = \mathrm{n_A^{max}} = \mathrm{n_S^{max}} = 200$.

Table 1
Settings of the BoomerAMG preconditioner [60]. These choices are made based on the balance of robustness and efficiency. Interested readers are referred to [61,62] for a discussion of these options.

Cycle type	V-cycle
Coarsening method	HMIS
Interpolation method	Extended method (ext + i)
Smoother	Hybrid Gauss–Seidel
Truncation factor for the interpolation	0.3
Threshold for being strongly connected	0.5
Maximum number of elements per row for interp.	5
The number of levels for aggressive coarsening	2

On the outer level, we invoke the FGMRES(m) to minimize the residual for $\boldsymbol{\mathcal{A}x} = \boldsymbol{r}$, and, unless otherwise specified, we fix m as 200, the absolute tolerance δ^a as 10^{-50}, and the maximum number of iterations $\mathrm{n}^{\max}$ as 200. With these settings, the stopping criterion is completely determined by the relative tolerance δ^r.

Remark 6. In the construction of the $\mathcal{P}_{\mathrm{SCR}}$, the full factorization of $\boldsymbol{\mathcal{A}}$ is utilized. This indicates that the algebraic representation of $\mathcal{P}_{\mathrm{SCR}}$ is identical to $\boldsymbol{\mathcal{A}}$, while the actual definition of $\mathcal{P}_{\mathrm{SCR}}$ is determined by the choices of the intermediate and inner solvers in Algorithm 3. In the meantime, we note that it is possible to use part of the block factorization to devise a preconditioner [63–65]. For instance, assuming exact arithmetic, using $\boldsymbol{\mathcal{D}}$ as the preconditioner leads to convergence in 4 iterations; using $\boldsymbol{\mathcal{DU}}$ guarantees convergence within 2 iterations; the full factorization warrants convergence in 1 iteration. In our experience, using the full factorization effectively mitigates the number of solving the Schur complement, which is the most expensive part in Algorithm 3. The price paid is additional iterations needed in the solution procedure associated with $\mathbf{A}$, which can be conveniently and efficiently addressed by AMG or even simpler candidates (see the discussion in Section 4.4).

Remark 7. When using higher-order elements, it is often economical to avoid explicitly constructing the tangent matrix. Instead, the action of the tangent matrix on a vector can be achieved through a matrix-free manner using, e.g., the partial assembly approach [66] or the Newton–Raphson–Krylov technique [67]. In the meantime, a matrix can be created using low-order elements on the same mesh to construct preconditioners. Although this work focuses on the low-order finite element discretization of the incompressible Navier–Stokes equations, it may serve as a competitive candidate for higher-order methods [68–70].

3.2. The SIMPLE preconditioner

The SIMPLE preconditioner is an algebraic analogue of the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) [71]. Together with its variants, it is among the most popular choices for CFD [11,22], FSI [72], and other multiphysics problems [73,74]. It is partly inspired from the block factorization (3.1) and can be represented as follows,

$$\mathcal{P}_{\mathrm{SIMPLE}} := \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{O} \\ \mathbf{O} & \hat{\mathbf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & (\mathrm{diag}\,(\mathbf{A}))^{-1}\,\mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{A}\mathrm{diag}\,(\mathbf{A})^{-1}\,\mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}.$$

Remember that $\hat{\mathbf{S}} := \mathbf{D} - \mathbf{C}\,(\mathrm{diag}\,(\mathbf{A}))^{-1}\,\mathbf{B}$ is a sparse approximation of $\mathbf{S}$. The purpose of choosing $(\mathrm{diag}\,(\mathbf{A}))^{-1}$ in the upper diagonal matrix is to ensure that the mass equation is not perturbed [10,75]. The action of $\mathcal{P}^{-1}_{\mathrm{SIMPLE}}$ on a vector can be implemented in a way similar to Algorithm 3 with modifications to the steps 3 and 4. Roughly speaking, one may view the SIMPLE preconditioner as a simplification of the SCR approach without invoking the inner solver. Similar to the SCR preconditioner, one may invoke a Krylov subspace method preconditioned with multigrid [10,11] or domain decomposition preconditioners [72] on the intermediate level to define the action of $\mathcal{P}^{-1}_{\mathrm{SIMPLE}}$. Since the actual algebraic definition of the SIMPLE preconditioner also varies over successive iterations, one still has to use the FGMRES method as the outer solver. In this work, the settings of the outer and intermediate solvers in $\mathcal{P}_{\mathrm{SIMPLE}}$ always follow the settings used in $\mathcal{P}_{\mathrm{SCR}}$ in each specific example.

3.3. The bi-partitioned iterative algorithm

In the current `svSolver`, the linear system of equations is solved by the BIPN algorithm [32] used in conjunction with a specifically designed preconditioner for the resistance-type boundary condition [31]. The action of this preconditioner on a vector is achieved through a five-step procedure similar to the one given in Algorithm 3, with the following differences. First, in step 3, the following equation is solved by an unpreconditioned conjugate gradient method,

$$\tilde{\mathbf{S}}\boldsymbol{x}_p = \boldsymbol{r}_p, \tag{3.6}$$

$$\tilde{\mathbf{S}} := \mathbf{D} - \mathbf{B}^T \left[(\mathrm{diag}\,(\mathbf{F}))^{-1} - \alpha_f \gamma \Delta t_n \sum_{k=1}^{\mathrm{n_{out}}} \mathfrak{m}^k \frac{\boldsymbol{b}^k \boldsymbol{b}^{kT}}{1 + \alpha_f \gamma \Delta t_n \mathfrak{m}^k \|\boldsymbol{b}^k\|_{l^2}^2} \right] \mathbf{B}, \tag{3.7}$$

$$\boldsymbol{b}^k := (\mathrm{diag}\,(\mathbf{F}))^{-1}\, \boldsymbol{a}^k. \tag{3.8}$$

The definition of $\tilde{\mathbf{S}}$ is inspired from the Sherman–Morrison formula for inverting a matrix with rank-one modifications [76]. Second, on the outer level, the `svSolver` solves a least-square problem to minimize the residual by inverting a normal matrix.

It has been demonstrated that this preconditioner is an effective candidate for hemodynamic simulations since the contributions from the outflow boundary conditions are taken into account. However, this method is not without shortcomings. Firstly, the equation associated with $\tilde{\mathbf{S}}$ is solved without a preconditioner because the rank-one matrices in $\tilde{\mathbf{S}}$ are not explicitly constructed. The rank-one matrix is a dense block matrix defined on each outlet surface, necessitating a non-trivial sparsity pattern in the matrix allocation stage. It is quite inconvenient to assemble this matrix in a standard finite element code. However, the action of a rank-one matrix on a vector can be efficiently calculated as a vector inner product in the parallel setting [54, p. 3547], which leads to a matrix-free definition of the matrices $\mathbf{A}$ and $\tilde{\mathbf{S}}$. Using a matrix-free algorithm can be convenient and efficient for small- or medium-sized problems. However, without using a preconditioner, we may reasonably expect a loss of scalability in the performance of `svSolver` as the problem size gets larger. Secondly, similar to $\mathcal{P}_{\mathrm{SIMPLE}}$, the convection information is still missing in the definition of this preconditioner, rendering the method ineffective for strong convection problems. Thirdly, the usage of a normal matrix to minimize the residual will inevitably lead to a nearly singular problem with an increased size of the subspace (see Fig. 7). Thereby the outer solver is typically limited to at most ten iterations in `svSolver` [33]. This signifies a numerical robustness issue and poses a limit on the accuracy that the current `svSolver` can attain. Lastly, the `svSolver` underperforms for FSI problems using the coupled momentum method. The reason is apparently that the wall stiffness matrix is not reflected in $\tilde{\mathbf{S}}$. Indeed, for more complicated problems, the definition of $\mathbf{A}$ may involve contributions from different physics. This motivates us to consider using the matrix-free algorithm (i.e. Algorithm 2) to attain a better representation of the Schur complement.

Remark 8. There exist variants of the SIMPLE preconditioner, such as the SIMPLEC preconditioner, that incorporate more information from off-diagonal entries [77]. It has been suggested that the SIMPLEC preconditioner often serves as a competitive candidate for CFD simulations [11]. Yet, in this study, a straightforward application of the SIMPLEC preconditioner for $\mathbf{A}$ often underperforms in comparison with the SIMPLE preconditioner, due to the weighted sum of rank-one matrices. The preconditioner $\tilde{\mathbf{S}}$ above provides an appropriate way of incorporating the information from the rank-one matrices. It is tempting to consider replacing the diagonal of $\mathbf{F}$ in (3.7)–(3.8) by the row sum of the absolute values of entries in $\mathbf{F}$, as an extension of the SIMPLEC preconditioner. Exploration of this idea remains a topic of our future work.

Remark 9. In Algorithm 3, $\mathbf{P}_{\mathrm{S}}$ is constructed based on $\hat{\mathbf{S}}$. If Eq. (3.4) is solved by applying the preconditioner once without entering into the Krylov iteration, the $\mathcal{P}_{\mathrm{SCR}}$ preconditioner reduces to the SIMPLE preconditioner $\mathcal{P}_{\mathrm{SIMPLE}}$. In this regard, the $\mathcal{P}_{\mathrm{SCR}}$ preconditioner can be viewed as a generalization of the conventional block preconditioner in that it uses the Krylov iteration to enhance the approximation of the Schur complement. It is therefore tempting to consider constructing $\mathbf{P}_{\mathrm{S}}$ based on a sparse matrix that incorporates more information from $\mathbf{S}$, such as $\tilde{\mathbf{S}}$ given above or the PCD preconditioner [24]. It is also worth reconsidering using the LSC preconditioner [23] in $\mathcal{P}_{\mathrm{SCR}}$, in the hopes of alleviating robustness issues of the LSC in the stabilized formulation [11].

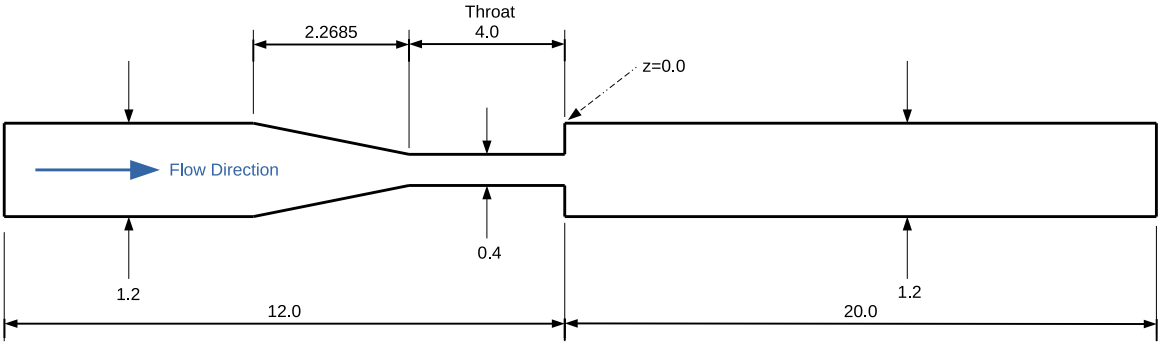


Fig. 3. Schematic representation of the cross section of the computational domain for the idealized medical device problem.

4. Numerical results

Numerical simulations reported in this work are all performed on the Stampede2 supercomputer at Texas Advanced Computing Center (TACC), using the Intel Xeon Platinum 8160 "Skylake" node. Each compute node contains 48 processors, with 2.1 GHz nominal clock rate, and 192 GB RAM. They are interconnected by a 100 Gb/s Intel Omni-Path network. More technical details about the Stampede2 supercomputer can be found in its user guide [78].

In this work, we are primarily concerned with blood flow simulations. The fluid density is fixed to be 1.065 g/cm^3 and the dynamic viscosity is fixed to be 0.035 poise for all simulations. On the inlet surface Γ_{in}, we always apply a parabolic velocity profile for $\boldsymbol{v}_{\text{in}}$ based on a prescribed flow rate $\mathcal{Q}$.

4.1. FDA idealized medical device benchmark

In the first numerical example, we consider a benchmark designed by the US Food and Drug Administration (FDA) to examine the CFD code performance and validate the code by comparing with in vitro experiments. The geometry of the idealized medical device is illustrated in Fig. 3. We follow the practice adopted in [79], in which the computational domain is chosen to be 32 cm long. Based on a parabolic velocity profile, we may calculate the maximum velocity on the inlet $\boldsymbol{v}^{\text{i}}_{\text{max}}$ as well as at the throat region $\boldsymbol{v}^{\text{t}}_{\text{max}}$ from a given value of $\mathcal{Q}$. Zero traction boundary condition is applied on the outlet surface. The element size Δx here is defined as the circumscribing sphere's diameter. In this example, we choose $\varrho_\infty = 1.0$ in the generalized-α method to mitigate the numerical dissipation effect, which is critical in this example. With this choice, the time-stepping method reduces to the mid-point rule. The Courant number Cr reported in this section is calculated based on the minimum mesh size Δx_{min} and $\boldsymbol{v}^{\text{t}}_{\text{max}}$.

4.1.1. Solver validation

Before investigating this problem, the CFD solver has been verified using the manufactured solution method, and optimal convergence rates for the velocity and pressure have been observed. Here, we perform a suite of simulations to further validate it. An isotropic unstructured mesh is generated by Gmsh [80] using the frontal algorithm. The mesh consists of 28.8 million linear tetrahedral elements and 4.94 million vertices. In particular, we do not perform local mesh refinement near the sudden expansion region to ease the reproducibility of the results and to give a fair evaluation of the predictive capability of our CFD code. The minimum and maximum element sizes of the mesh are $\Delta x_{\text{min}} = 0.014$ cm and $\Delta x_{\text{max}} = 0.069$ cm, respectively. In Table 2, the volumetric flow rates of the inflow, the flow speed, the corresponding Reynolds numbers, and the time step sizes are summarized. With the varying values of $\mathcal{Q}$, the Reynolds numbers at the throat take the values of 500, 3500, and 5000. These values correspond to the laminar, transitional, and fully turbulent regimes, thereby offering a wide range of flow characteristics for code validation. The inflow velocity profile is perturbed by a random velocity field with zero mean and a standard deviation 1% of the mean axial velocity [79,81]. We start the simulation with zero velocity and the inflow rate is gradually increased to reach the target values. In Fig. 4, the instantaneous velocity magnitudes for the three cases

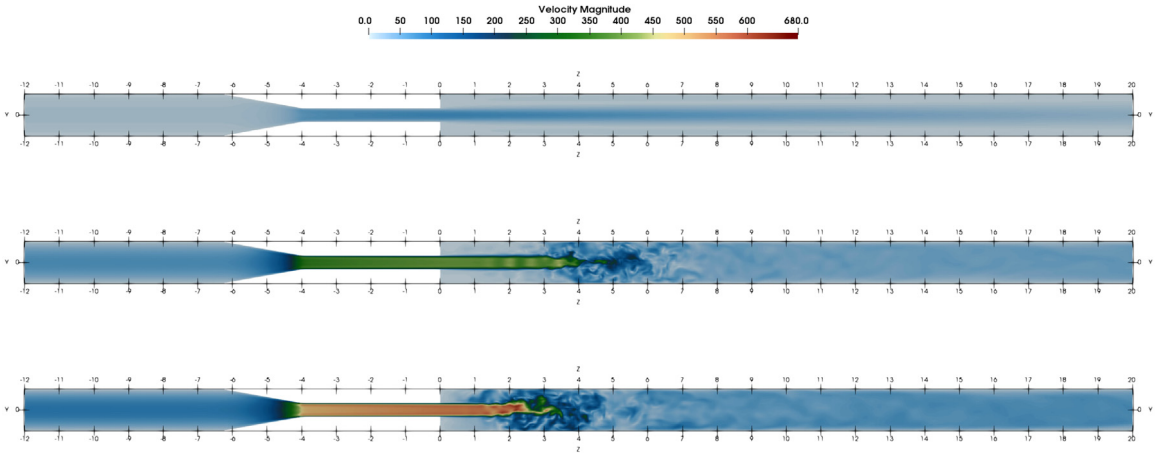


Fig. 4. Instantaneous velocity magnitude in cm/s for $\mathrm{Re}^{\mathrm{t}} = 500$ (top), $\mathrm{Re}^{\mathrm{t}} = 3500$ (middle), and $\mathrm{Re}^{\mathrm{t}} = 5000$ (bottom).

Table 2
The flow rates $\mathcal{Q}$, maximum velocities in the inlet and the throat, the corresponding Reynolds numbers, and the time step sizes used in the simulations. The maximum velocities are calculated based on a parabolic profile.

$\mathcal{Q}$ [cm^3/s]	$v^{\mathrm{i}}_{\max}$ [cm/s]	Re^{i}	$v^{\mathrm{t}}_{\max}$ [cm/s]	Re^{t}	Δt [s]
5.2062×10^0	9.21×10^0	167	8.29×10^1	500	1.0×10^{-4}
3.6444×10^1	6.45×10^1	1167	5.80×10^2	3500	5.0×10^{-5}
5.2062×10^1	9.21×10^1	1667	8.29×10^2	5000	2.5×10^{-5}

Table 3
The time step size Δt (in the unit of seconds) for different Courant number Cr and Reynolds number Re^{t} in the simulations of Section 4.1.2.

Cr	$\mathrm{Re}^{\mathrm{t}} = 500$	$\mathrm{Re}^{\mathrm{t}} = 3500$	$\mathrm{Re}^{\mathrm{t}} = 5000$
0.5	3.69×10^{-4}	5.28×10^{-5}	3.69×10^{-5}
2.0	1.48×10^{-3}	2.11×10^{-4}	1.48×10^{-4}

considered are depicted over a cut in the middle of the geometry. For the case of $\mathrm{Re}^{\mathrm{t}} = 500$, the flow reaches a steady state and shows an axially symmetric pattern. For the cases of $\mathrm{Re}^{\mathrm{t}} = 3500$ and 5000, the flows are highly unsteady, with an anticipated jet break point in the sudden expansion region. In Fig. 5, the time-averaged axial velocities along the nozzle centerline are compared with experimental data obtained by different laboratories [82]. As can be observed from the figures, the simulation results show good agreement with the experimental data. This validation is preliminary, and ongoing efforts are still needed to quantify the impact of inflow fluctuations and numerical settings on the reliability of the CFD solver's predictions.

4.1.2. Algorithm robustness

In this test, we examine the solver performance with different Reynolds numbers. The spatial mesh contains 7.82×10^5 tetrahedral elements and 1.43×10^5 vertices. Correspondingly, the system of equations involves 5.71×10^5 unknowns. The minimum and maximum element sizes of this mesh are 0.06 cm and 0.29 cm, respectively. We choose to simulate with two different Courant numbers, $\mathrm{Cr} = 0.5$ and 2.0, and the time step sizes are summarized in Table 3. The simulations are performed on a single node with 6 CPUs assigned. We choose $\delta^r = 10^{-8}$ for the FGMRES iterations. For the nested block preconditioner, we set $\delta^r_A = \delta^r_S = \delta^r_I$ and vary their values from 10^{-8} to 10^{-2}. For the SIMPLE preconditioner, we investigated two choices: $\delta^r_A = \delta^r_S = 10^{-8}$ and 10^{-2}. For comparison purposes, we also applied the restricted additive Schwarz method implemented in PETSc [83] with the incomplete LU factorization as the subdomain solver. The convergence histories of this set of simulations are depicted in Fig. 6.

There are several salient features of the solver performance that can be summarized from Fig. 6. First, the nested block preconditioner always shows a faster rate in residual reduction. For all cases considered, the proposed

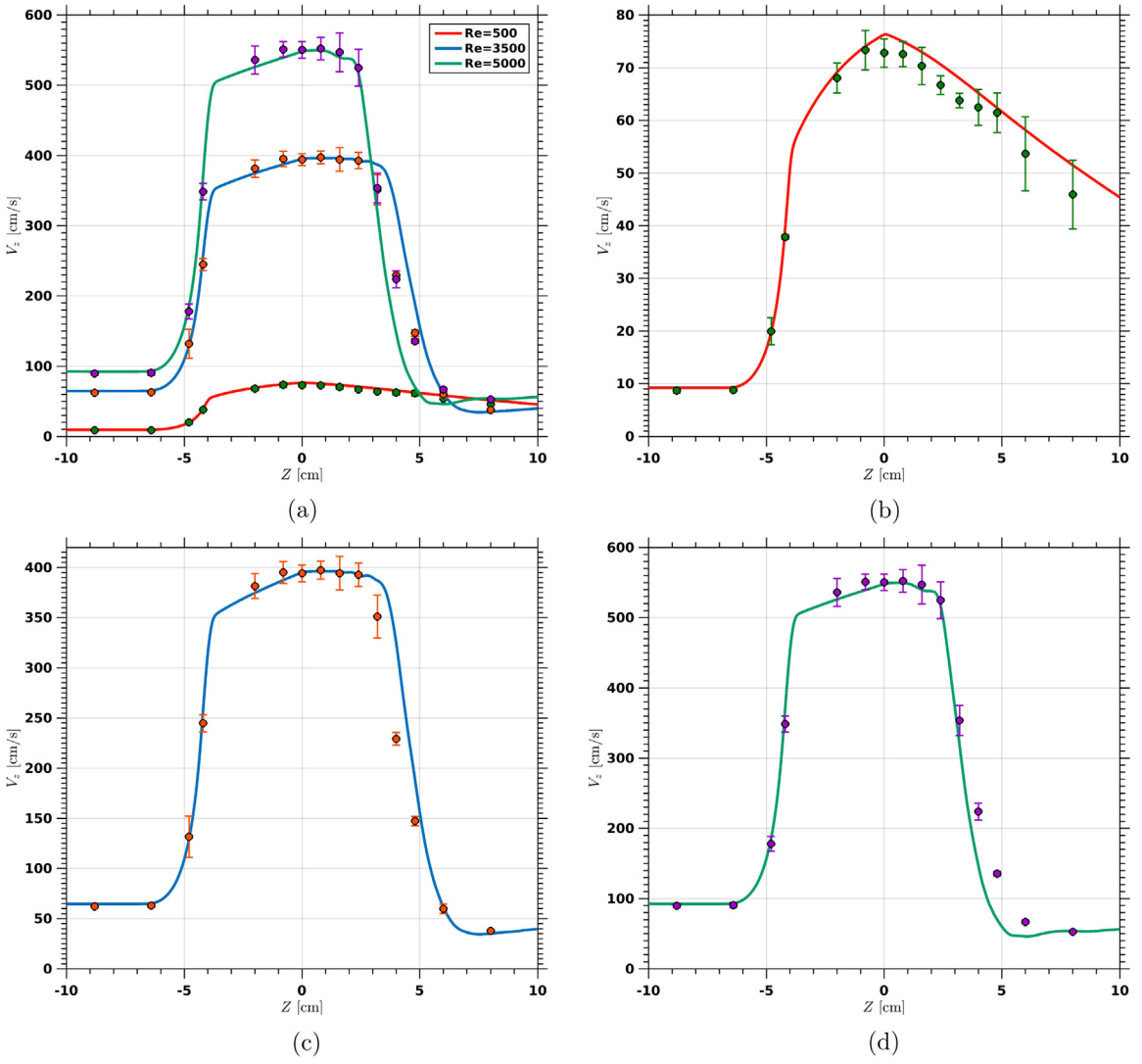
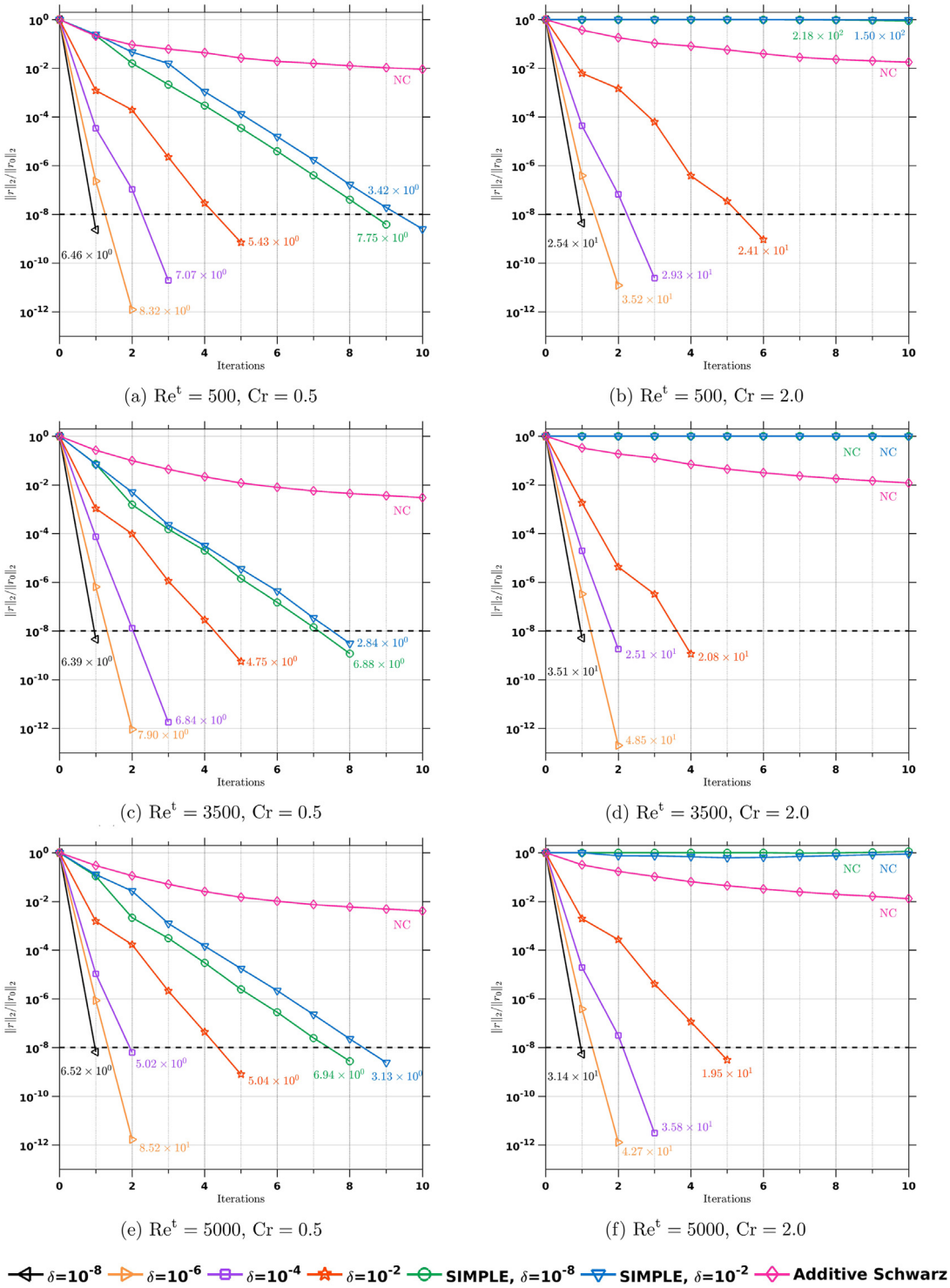


Fig. 5. The axial velocity along the nozzle centerline for all three Reynolds numbers (a). Detailed views of the velocity are given for $\mathrm{Re}^{\mathrm{t}} = 500$ (b), 3500 (c), and 5000 (d). Solid lines are time-averaged velocities from the CFD simulations, and the symbols represent experimental means with 95% confidence interval [82].

method requires at most 6 iterations to achieve convergence. In terms of the CPU time, we found that a looser tolerance for the intermediate and inner solvers always gives more efficient performance. Indeed, for the nested block preconditioner, a significant amount of computing time is spent on the matrix-free approximation of the Schur complement. A looser tolerance can surely reduce the computational burden from that part. This suggests that, in practice, one is advised to flexibly tune the tolerances for δ_A^r, δ_S^r, and δ_I^r to balance the solver robustness and the cost per iteration (see, e.g., the discussion in Section 5.1.1 of [41]). Also, in our experience, AMG is not the most cost-effective approach for solving $\mathbf{A}$, especially in the inner solver. Using a 'lightweight' preconditioner may further reduce the overhead.

Second, the SIMPLE preconditioner achieves convergence within ten iterations for all three cases when $\mathrm{Cr} = 0.5$. The choice of the intermediate solver accuracy does not have a significant impact on the overall convergence rate. The SIMPLE preconditioner with the intermediate solver accuracy 10^{-2} gives the fastest performance for all three cases of $\mathrm{Cr} = 0.5$. However, its drawback is demonstrated when the Courant number is greater than 1. When $\mathrm{Cr} = 2.0$, it only achieves convergence for the case of $\mathrm{Re}^{\mathrm{t}} = 500$, with around 130 iterations. For larger Courant



(a) $\mathrm{Re}^t = 500$, $\mathrm{Cr} = 0.5$

(b) $\mathrm{Re}^t = 500$, $\mathrm{Cr} = 2.0$

(c) $\mathrm{Re}^t = 3500$, $\mathrm{Cr} = 0.5$

(d) $\mathrm{Re}^t = 3500$, $\mathrm{Cr} = 2.0$

(e) $\mathrm{Re}^t = 5000$, $\mathrm{Cr} = 0.5$

(f) $\mathrm{Re}^t = 5000$, $\mathrm{Cr} = 2.0$

Fig. 6. Convergence history for $\mathrm{Cr} = 0.5$ (left column) and 2.0 (right column) for $\mathrm{Re}^t = 500$ (top row), $\mathrm{Re}^t = 3500$ (middle row), and $\mathrm{Re}^t = 5000$ (bottom row). The horizontal dashed black line indicates the prescribed stopping criterion for the relative error, which is 10^{-8} here. The numbers indicate the CPU time for the linear solver in the unit of seconds. NC stands for no convergence within the prescribed number of iterations. In the legend, δ represents the relative tolerance used in the intermediate and inner solvers.

Table 4
The strong scaling performance for the FDA idealized medical device benchmark. In the table, T_A and T_L represent the timings for the matrix assembly and the linear solver, respectively; the efficiency is computed based on the total runtime.

Proc.	T_A (s)	T_L (s)	Total (s)	Efficiency
2	2.24×10^3	3.33×10^3	5.64×10^3	100%
4	1.25×10^3	1.74×10^3	3.03×10^3	93%
8	6.15×10^2	9.05×10^2	1.54×10^3	92%
16	2.91×10^2	4.78×10^2	7.80×10^2	91%
32	1.25×10^2	2.39×10^2	3.72×10^2	95%
64	6.38×10^1	1.42×10^2	2.11×10^2	84%
128	3.39×10^1	7.80×10^1	1.17×10^2	75%
256	1.84×10^1	5.53×10^1	8.04×10^1	55%

and Reynolds numbers, the matrix $\mathbf{A}$ is no more diagonally dominated, and thus the SIMPLE preconditioner shows rather poor performance. As will be demonstrated in Section 4.2, it is quite sensitive to the resistance value as well. Although the SIMPLE preconditioner gives competitive performance under certain scenarios, its poor robustness limits the capability of the overall algorithm, considering the scheme adopted is fully implicit. The efficacy of the inner solver can be observed from Fig. 6(b), (d), and (e). Even a loose tolerance, such as $\delta_I^r = 10^{-2}$, significantly improve the robustness of the overall method.

Third, the additive Schwarz domain decomposition preconditioner cannot drive the relative error to 10^{-8} within the prescribed number of iterations (10000 here) for any of the six cases. It is also known that this preconditioner does not scale well. An increase in the problem size will further degrade its performance in terms of the iteration number and CPU time cost.

4.1.3. Parallel performance

In this section, we examine the proposed preconditioning technique by investigating its performance under the parallel setting. The stopping criterion for the FGMRES iteration is $\delta^r = 10^{-6}$, and the tolerances for the intermediate and inner solvers are $\delta_A^r = \delta_S^r = \delta_I^r = 10^{-3}$.

Fixed-size scalability. In this test, we investigate the fixed-size scalability of the algorithm. Rather than starting from a zero solution, we prepared a developed flow profile at $\mathrm{Re}^t = 3500$. The spatial mesh contains 5.71×10^6 tetrahedral elements and 1.00×10^6 vertices, and the overall system of equations involves 4×10^6 unknowns. Then the same mesh is re-partitioned based on the given number of processors. The prepared flow profile is mapped to the new partitioned mesh to serve as the initial condition. The time step size is fixed to be 5×10^{-5} s and the problem is integrated for 40 time steps. Due to the hierarchical architecture of Stampede2, the simulations are performed with one CPU assigned in each node, which means the number of processors reported in Table 4 equals the number of nodes. In doing so, the MPI messages are communicated purely through the Intel Omni-Path network. In Table 4, we report the timings as well as the speed-up efficiency based on the total time as the number of CPUs increases. It can be observed that the scaling shows nearly optimal speed-up efficiency for a wide range of processor counts for the proposed nested block preconditioner.

Isogranular scalability. In the next test, we study the isogranular or weak scalability of the proposed preconditioning technique. We generated three sets of isotropic unstructured meshes and partitioned the mesh with the goal of maintaining a constant number of unknowns assigned to each CPU. The initial conditions are prepared by gradually increasing the flow rate to reach the targeted values in one second and maintaining the targeted flow rates for another second. Then the obtained solutions are used as the initial condition for the scaling test. For each different run, the time step size is chosen based on the Courant number, and we select to simulate with $\mathrm{Cr} = 0.5$, 1.0, and 2.0. The statistics of the solver performance are collected for 20 time steps (Table 5). It can be observed that the averaged number of iterations remains between 2 and 3 for different problem sizes and the Courant numbers. For a given Courant number, the averaged CPU time grows mildly (three to four times longer with a sixty-four-fold increase of the problem size).

Table 5
Comparison of the averaged iteration counts $\bar{n}$ and the CPU time for solving the linear system $\bar{T}_L$ in seconds for the nested block preconditioner. The values of $\Delta x_{\min}$ for the three meshes are 6.8×10^{-2}, 3.1×10^{-2}, 1.4×10^{-2} cm, respectively.

n_{en}	n_{eq}	Proc.	Cr = 0.5		Cr = 1.0		Cr = 2.0	
			$\bar{n}$	$\bar{T}_L$ (s)	$\bar{n}$	$\bar{T}_L$ (s)	$\bar{n}$	$\bar{T}_L$ (s)
$\mathrm{Re}^t = 500$								
4.20×10^5	3.10×10^5	6	2.45	1.82	2.41	2.12	2.47	4.20
3.65×10^6	2.48×10^6	48	2.50	2.83	2.60	3.34	2.89	9.34
2.88×10^7	1.98×10^7	384	2.90	6.97	2.95	7.78	3.02	15.05
$\mathrm{Re}^t = 3500$								
4.20×10^5	3.10×10^5	6	2.38	1.99	2.36	2.23	2.13	4.23
3.65×10^6	2.48×10^6	48	2.62	3.20	2.31	3.11	2.27	6.51
2.88×10^7	1.98×10^7	384	2.78	6.95	2.58	6.96	2.45	11.58
$\mathrm{Re}^t = 5000$								
4.20×10^5	3.10×10^5	6	2.35	1.83	2.22	2.12	2.21	4.40
3.65×10^6	2.48×10^6	48	2.76	3.24	2.52	3.31	2.33	6.64
2.88×10^7	1.98×10^7	384	2.83	6.96	2.52	7.17	2.41	13.41

4.2. A cylindrical model with resistance boundary condition

In this example, we investigate the preconditioner performance with a resistance boundary condition (2.12). The geometry of this problem can be found as the first example in the simulation guide of SimVascular [33], which is a straight cylinder with a radius 2 cm and length 30 cm. On the inlet surface, we apply a parabolic velocity profile with a flow rate $\mathcal{Q} = 100$ cm^3/s. On the outlet boundary surface, we apply the resistance boundary condition (2.12) with varying values of R.[1] The mesh is generated using TetGen [84], and it consists of 1.74×10^6 isotropic tetrahedral elements and 2.88×10^5 vertices. The minimum element size is $\Delta x_{\min} = 0.1$ cm. For comparison, we also simulate the same problem using `svSolver` [33]. The solution method of `svSolver` is based on the BIPN iterative algorithm introduced in Section 3.3. *We caution the readers that the comparisons made between the proposed method and the `svSolver` are not apples-to-apples because the numerical formulation and the resulting matrix problem are different.* Comparisons are made merely to give overall evaluations of the two existing solvers under realistic settings.

In the first study, we choose $\Delta t = 6.3 \times 10^{-3}$ s for the simulation so that the Courant number Cr = 1.0 based on the maximum flow speed and $\Delta x_{\min}$. The simulations are performed with 24 CPUs. The resistance value R varies from 10^2 g/(s cm^4) to 10^5 g/(s cm^4). Again, we choose $\delta^r = 10^{-8}$ as the stopping criterion. In the SIMPLE and nested block preconditioners, we choose $\delta_A^r = \delta_S^r = 10^{-4}$, and the value of δ_I^r varies from 10^{-4} to 10^{-1}. We also simulated the same problem using `svSolver` with tolerances on the momentum and continuity equations fixed to 10^{-4}. The convergence histories and the CPU time are depicted in Fig. 7. In all cases considered, `svSolver` shows a rapid reduction of the residual in the first a few iterations and stagnates when the relative error is driven to approximately 10^{-4}. As one proceeds in the linear iteration, the normal matrix (see Eq. (45) in [32]) in the BIPN algorithm becomes nearly singular and causes divergence eventually. As we increase the resistance value, the SIMPLE preconditioner is also observed to stagnate. With an inner solver invoked, the proposed algorithm robustly drives the residual to the prescribed tolerance. For example, when R = 10^5, the algorithm converges in 64 iterations with $\delta_I^r = 10^{-1}$. This fact again indicates that a slight improvement of the numerical representation of the Schur complement may significantly improve the convergence rate of the overall iterative algorithm.

In the second study, we choose to compare the impact of the two solvers on the overall time-stepping algorithms for 30 time steps. The resistance value is fixed to be 1333 g/(s cm^4), leading to a pressure value of 100 mmHg at the outlet. The time step size is chosen to be $\Delta t = 3.15 \times 10^{-3}$ s. The tolerance δ^r is fixed to be 10^{-3}, with $\delta_A^r = 10^{-3}$ and $\delta_S^r = \delta_I^r = 10^{-2}$. We use the Jacobi preconditioner for the inner and intermediate solvers associated with $\mathbf{A}$, because this option is often more efficient than the AMG preconditioner (see also our discussion in Section 4.4). For `svSolver`, we limit the maximum number of iterations to 5 to avoid the numerical instability issue; the tolerances on momentum equations and the continuity equation are fixed to be 10^{-3} and 10^{-2}, respectively. Notice that in

[1] Notice that there is only one outlet surface $\Gamma_h = \Gamma_{\text{out}}^1$ in this case, and we neglect the superscript of R^1 for notational simplicity.

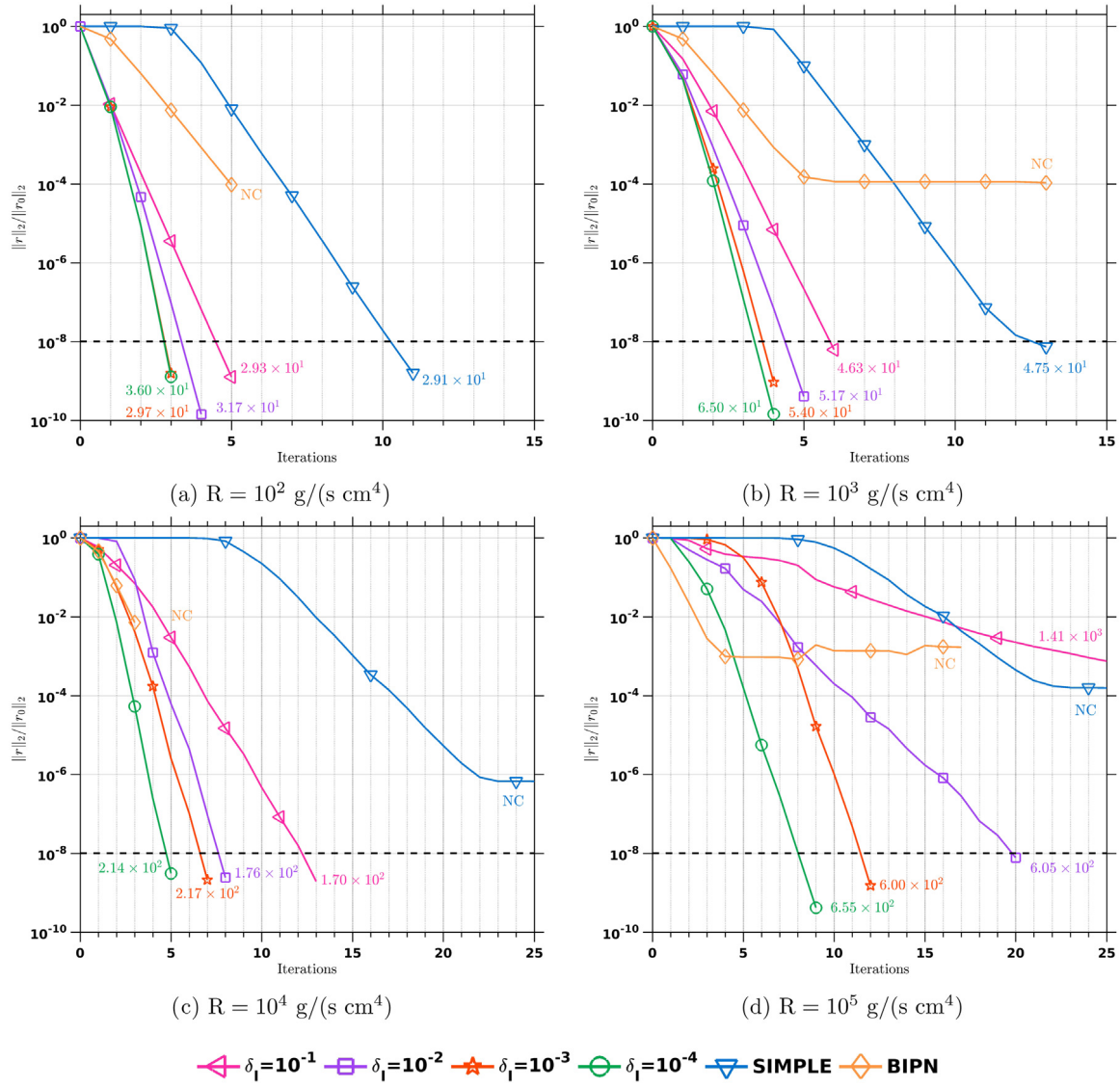


Fig. 7. Convergence history for different resistance values. The horizontal dashed black line indicates the prescribed stopping criterion for the relative error, which is 10^{-8} here. The numbers indicate the CPU time for the linear solver in the unit of seconds. NC stands for no convergence within the prescribed number of iterations.

`svSolver`, only the absolute tolerance is used as the stopping criterion. Hence, we make two sets of comparisons using $\text{tol}_\text{A} = 10^{-3}$ and 10^{-6} without invoking the relative tolerance. In Fig. 8, the CPU time and the number of Newton–Raphson iterations in the first 30 time steps are reported. It takes more iterations and CPU times in the first a few time steps for both solvers. After around the fifteenth time step, both solvers behave steadily over the remaining time steps. When $\text{tol}_\text{A} = 10^{-3}$, both solvers take approximately the same amount of CPU time for each time step after the twentieth time step. When $\text{tol}_\text{A} = 10^{-6}$, `svSolver` cannot give an accurate enough solution for the linear problem within its maximum number of iterations (5 here). This results in a higher amount of total computing cost because more multi-corrector steps have to be taken. The efficiency of the proposed algorithm will be further corroborated in the following section.

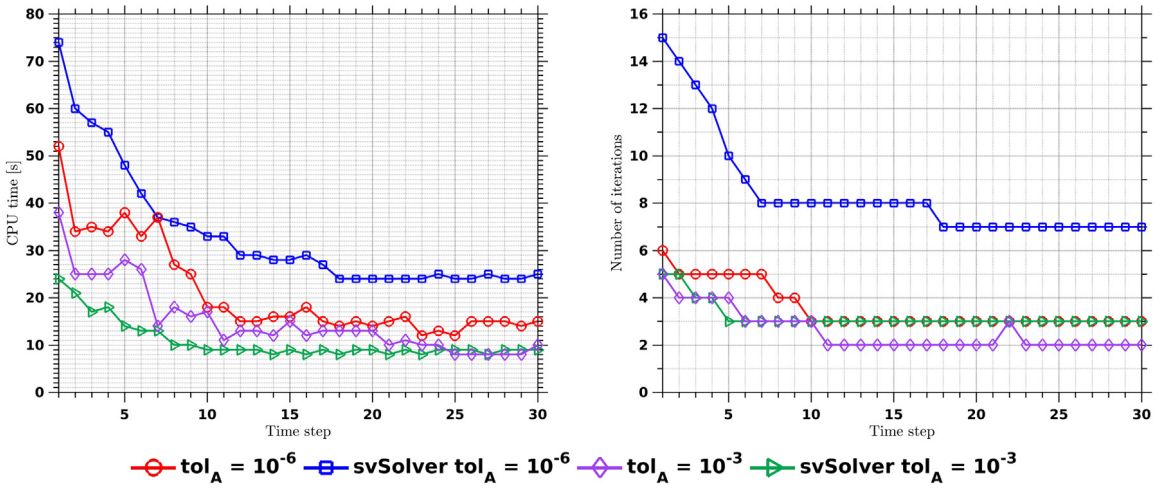


Fig. 8. Comparison of the CPU time (left) and the number of the multi-corrector iterations (right) in the first 30 time steps between the proposed algorithm and the `svSolver`.

4.3. Patient-specific hemodynamic simulation

In this section, we further examine the proposed method under physiologically realistic settings by considering two patient-specific examples: a diseased aortofemoral model and a healthy pulmonary model. The models and the clinical data are publicly available through the Cardiovascular and Pulmonary Model Repository [85]. The two cases are selected to cover the systemic and pulmonary circulations. The meshes for both examples are generated by Meshsim [86], with boundary layer meshes near the arterial wall. We simulate the problems using the nested block preconditioner with $\delta^r = 10^{-2}$, $\delta^r_A = 10^{-3}$, $\delta^r_S = \delta^r_I = 10^{-2}$, and $\mathrm{m_A} = \mathrm{m_S} = \mathrm{n_A^{max}} = \mathrm{n_S^{max}} = 20$. Similar to the previous example, we use the Jacobi preconditioner for the inner and intermediate solvers associated with $\mathbf{A}$. The same problems are also simulated with `svSolver`. In `svSolver`, the parameters adopt the default values except that the tolerances on momentum equations and the continuity equation are chosen to be 10^{-3} and 10^{-2}, respectively. The above parameters are chosen to achieve the fastest performances for the two solvers separately. Since the linear systems in the two solvers are not identical, we only aim at comparing the overall algorithm efficiency, with the same accuracy for solving the nonlinear equations.

4.3.1. Aortofemoral model

We first consider an aortofemoral model with an abdominal aortic aneurysm. The medical image and the volume rendering of the model are illustrated in Fig. 9(a). The volumetric flow rate used for prescribing the velocity on the inlet surface is illustrated in Fig. 9(b). The generated mesh consists of 8.80×10^6 tetrahedral elements and 1.61×10^6 vertices (Fig. 9(c)). The minimum element size is $\Delta x_{\min} = 9.21 \times 10^{-3}$ cm, and the time step size is chosen as $\Delta t = 4.16 \times 10^{-4}$ s. Notice that one cardiac cycle takes 0.832 s, and thence it requires 2000 time steps for simulating one cardiac cycle. The simulations are performed with 288 CPUs and simulated for 12 cardiac cycles. The instantaneous pressure, the velocity magnitude, the time-averaged wall shear stress, and the oscillatory shear index at the peak systole are depicted in Fig. 10. The performances of the proposed solution method and the `svSolver` are monitored for 30 time steps around peak systole and mid diastole (see Fig. 11). For one cardiac cycle, the proposed solution method takes 14803 s and `svSolver` takes 28200 s.

4.3.2. Pulmonary model

The pulmonary circulation has lower pressures than the systemic circulation, but with the same amount of flow going through an extensive and concentrated tree of branching arteries. To demonstrate the performance of the proposed method in the pulmonary circulation, a model was built from a healthy 20-month-old male, consisting of 772 outlets. This mesh consists of 2.61×10^7 tetrahedral elements and 4.95×10^6 vertices. Correspondingly,

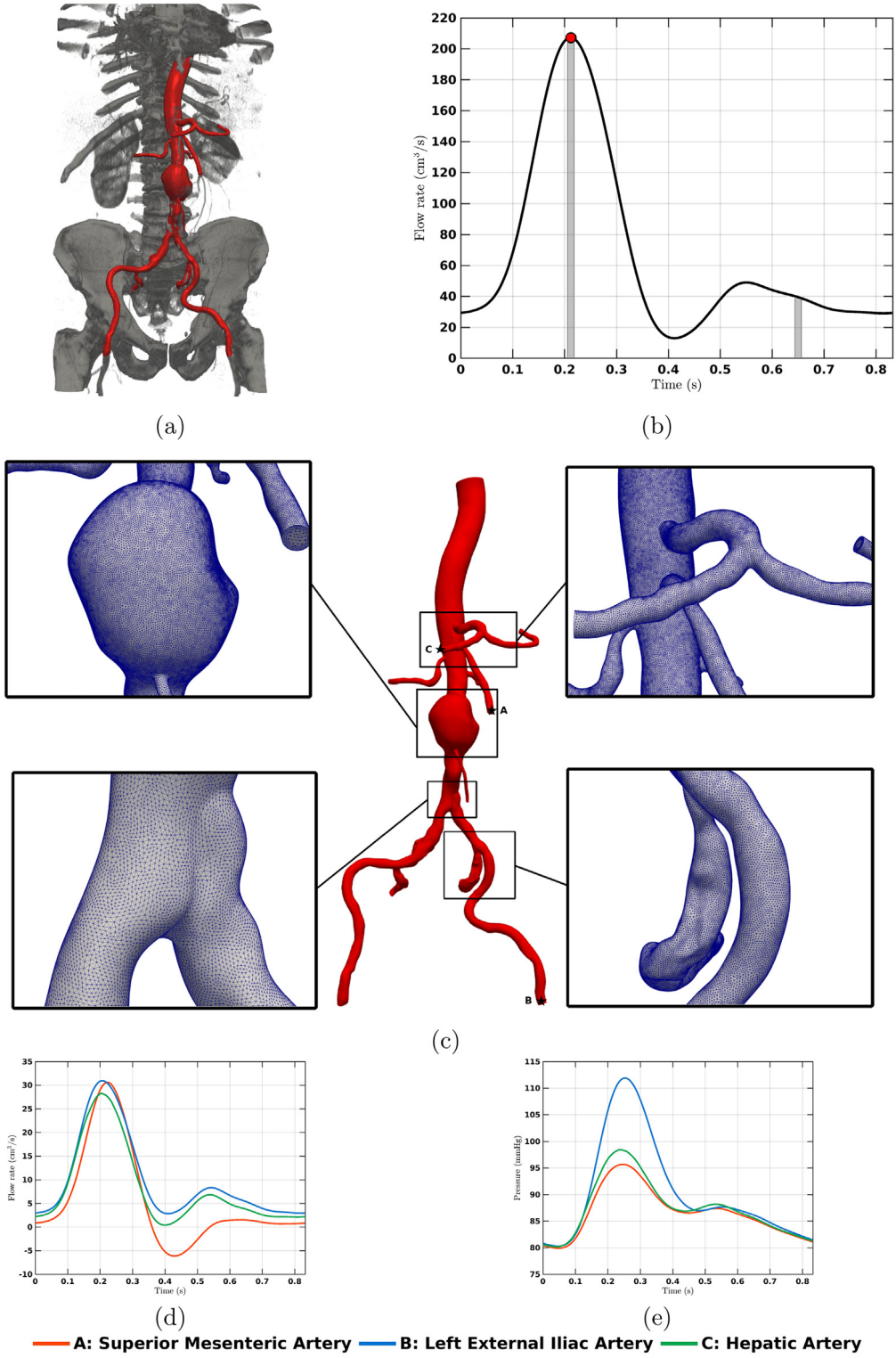


Fig. 9. (a) Volume rendering of the medical data. (b) Volumetric flow waveform used to prescribe the velocity on the inlet surface. The red dot represents the peak systole, and the gray regions represent the periods of time where we monitor the CPU time for the two different solvers. (c) The mesh of the aortofemoral model. The flow waveform (d) and pressure (e) on three outlets of the aortofemoral model.

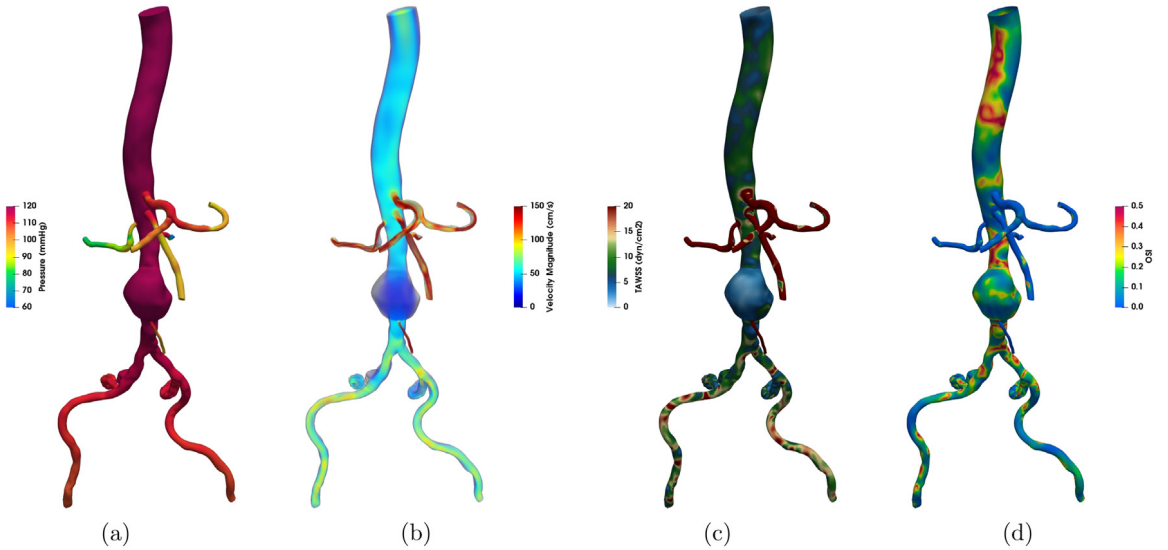


Fig. 10. The blood pressure (a), the velocity magnitude (b), the time-averaged wall shear stress (c), and the oscillatory shear index (d) of the aortofemoral model at peak systole.

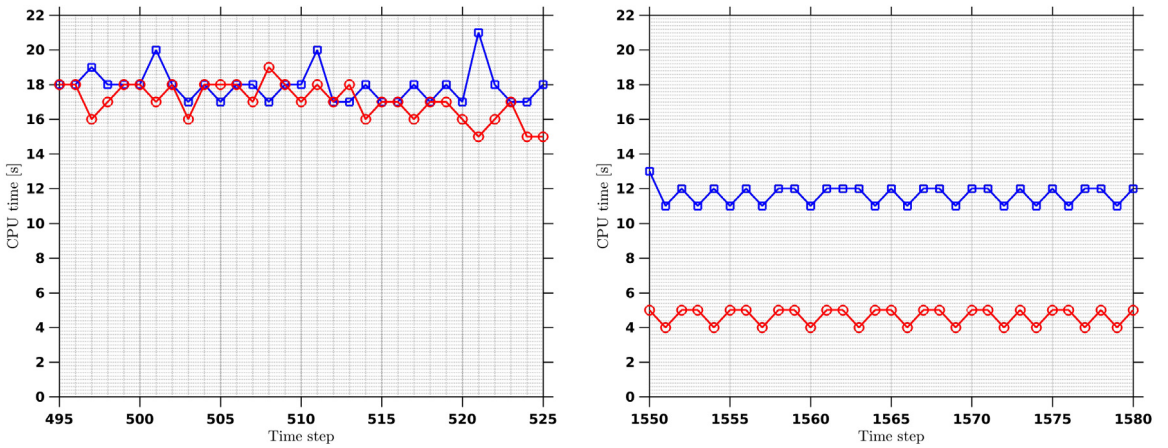


Fig. 11. Comparison of the CPU time for 30 time steps near the peak systole (left) and mid diastole (right). The curves represent the CPU time spent in each time step for the proposed method (red) and `svSolver` (blue). During the systole stage, the proposed solution method requires 4 Newton–Raphson iterations, while the `svSolver` reaches the maximum number of the Newton–Raphson iteration. During diastole, the proposed solution method converges in 2 Newton–Raphson iterations, and the `svSolver` needs 12 iterations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

there are about 20 million unknowns in the resulting linear system of equations. The minimum element size is $\Delta x_{\min} = 5.16 \times 10^{-3}$ cm. The time step size is fixed to be $\Delta t = 2.6995 \times 10^{-4}$ s. The medical image, volumetric flow rate, and the detailed views of the mesh are illustrated in Fig. 12. Notice that one cardiac cycle takes 0.5399 s and it requires 2000 steps in the time integration. The simulations are performed with 720 CPUs and simulated for 3 cardiac cycles. The instantaneous pressure, the velocity magnitude, the time-averaged wall shear stress, and the oscillatory shear index at the peak systole are depicted in Fig. 13. Performance of the proposed solution method and the `svSolver` is monitored for 30 time steps around peak systole and mid diastole (see Fig. 14). We notice that `svSolver` is faster than the proposed algorithm during the diastolic phase in this example. For one full cardiac cycle, the proposed method takes 14500 s and `svSolver` takes 19500 s. We also notice that the CPU time per time step is lower during the diastolic phase in both patient-specific simulations. This can be explained by the lower Reynolds

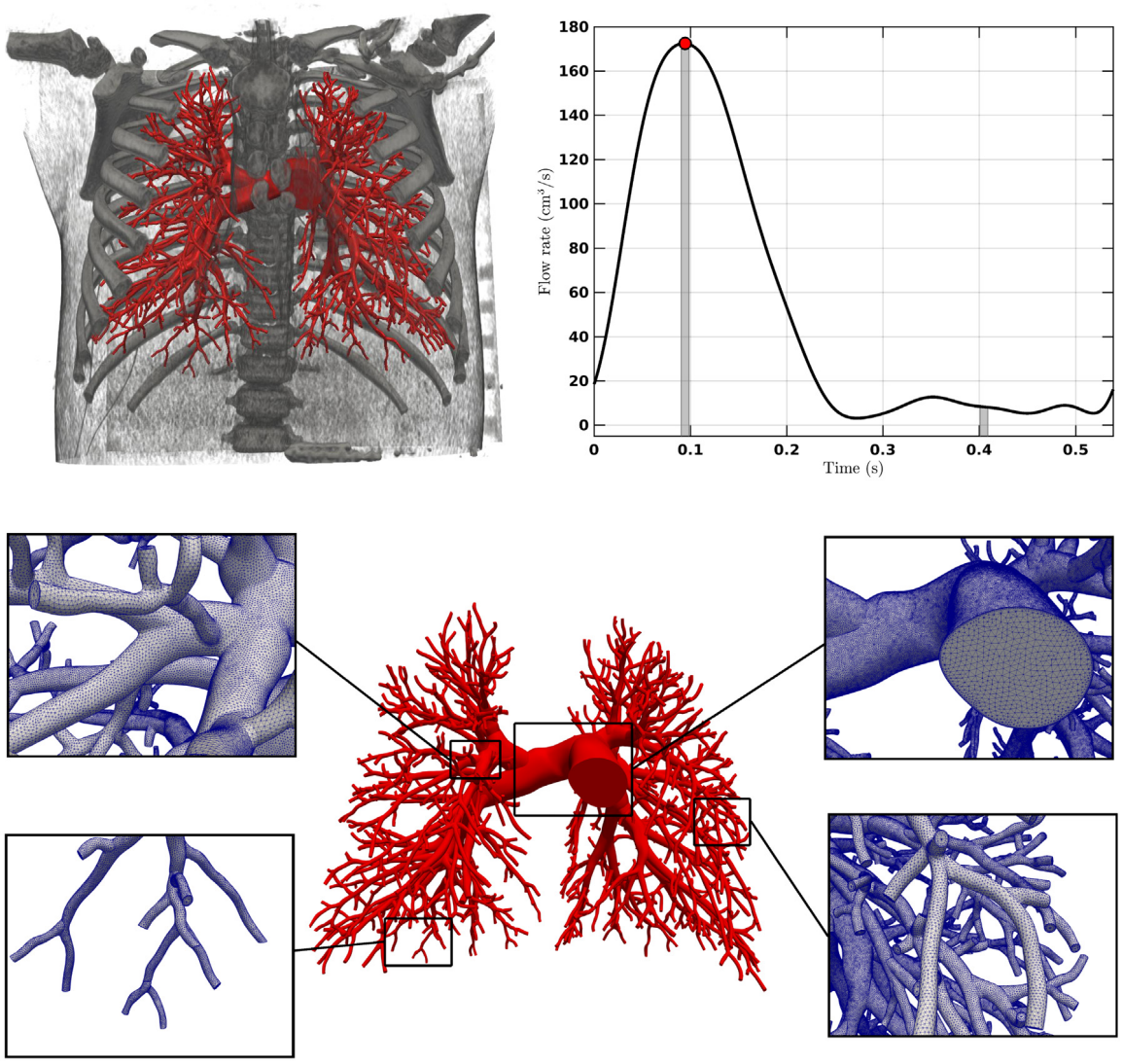


Fig. 12. (a) Volume rendering of the medical data. (b) Volumetric flow waveform used to prescribe the velocity on the inlet surface. The red dot represents peak systole. The gray regions represent the periods of time where we monitor the CPU time for the two different solvers in Fig. 14. (c) The mesh of the pulmonary model.

and Courant numbers in diastole. It is our experience that the conventional solver technique in the `svSolver` is non-robust, occasionally causing divergence during the systolic phase. As is revealed in Section 4.1.2, the three-level nested block preconditioner is less sensitive to these numbers and hence improves the overall algorithm robustness.

4.4. Additional discussion

As highlighted above, the inner solver plays a critical role in balancing the overall algorithm robustness and efficiency. The inner solver and the two solvers for (3.3 and (3.5) at the intermediate level are all associated with the block matrix **A**, and in this work, we apply the same solution strategy for these equations. It has been suggested to use AMG in the design of block preconditioners [21,87], which gives robust and scalable performances. However, the expensive setup and application phases of the AMG preconditioner result in a trade-off between scalability and

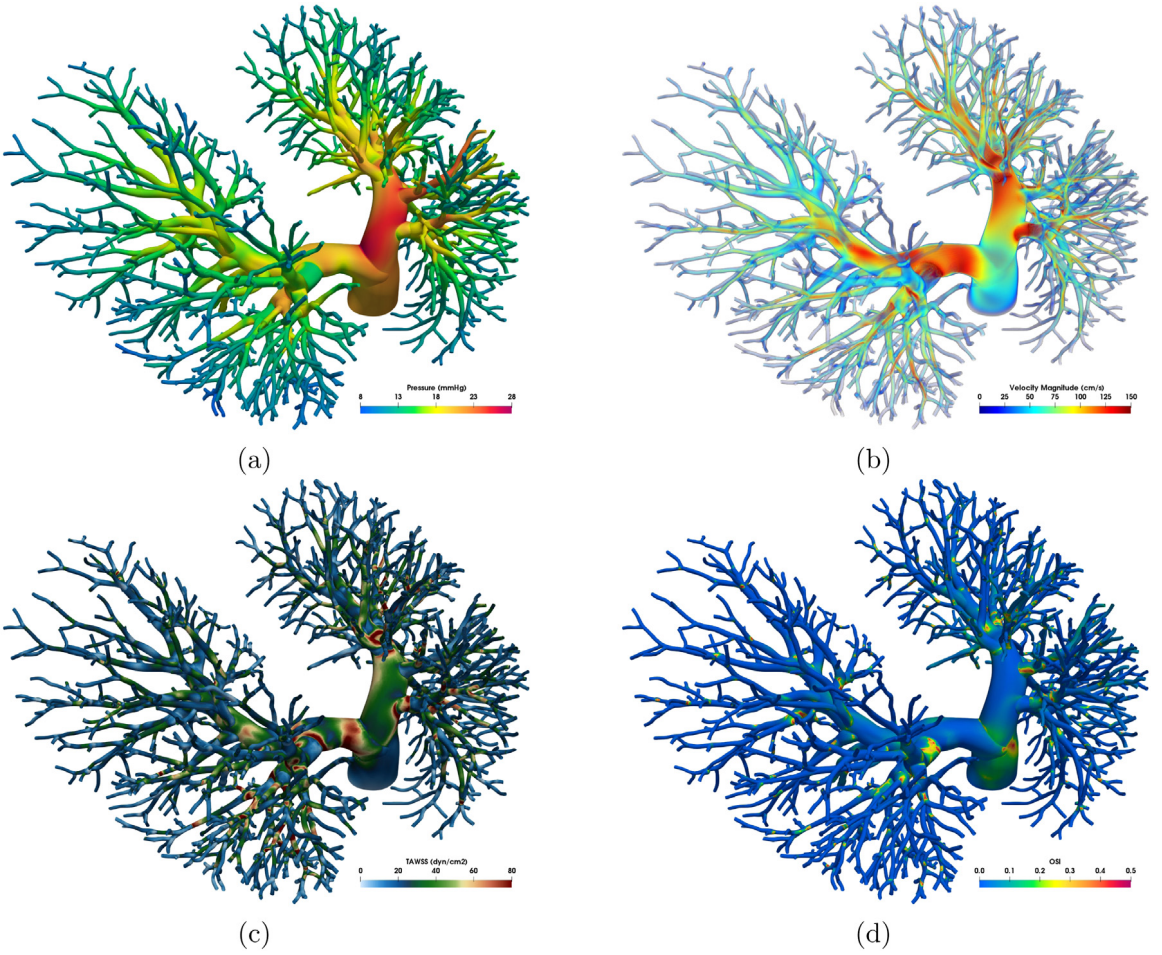


Fig. 13. The blood pressure (a), the volume rendering of the velocity magnitude (b), the time-averaged wall shear stress (c), and the oscillatory shear index (d) of the pulmonary model at peak systole.

efficiency. This can be particularly relevant when one simulates medium-sized problems when a scalable algorithm is not necessarily the most efficient one. For the problems considered in this work, we found that the AMG preconditioner is an expensive option for $\mathbf{A}$, and the Jacobi preconditioner is often efficient, even for the pulmonary model with around 20 million unknowns. In Fig. 15, we tested the AMG and Jacobi preconditioners for solving $\mathbf{A}$ with $\delta_A^r = 10^{-8}$ using both patient-specific models. Although the AMG preconditioner requires fewer iterations, it costs more CPU time compared to the Jacobi preconditioner. To further validate this observation, we monitor the solver performance using the FDA medical device model considered in Section 4.1. Three different meshes are considered and the tolerance is again set to be $\delta_A^r = 10^{-8}$. The results are summarized in Table 6. The Jacobi preconditioner is more efficient in all three meshes. Extrapolation suggests that the AMG preconditioner may be a more efficient option for finer meshes due to its better scalability. Notice that the value of δ_A^r rarely reaches 10^{-8} in practice. For looser tolerances, the advantage of the Jacobi preconditioner is more pronounced.

On the other hand we solve 3.4 with AMG because this step directly associates with the number of inner solver calls. We need a fast reduction rate when solving the Schur complement. Based on our experience, AMG outperforms other options in this regard. We also set $m_S = n_S^{max} = 20$ as a strategy to avoid spending too much time on solving the Schur complement.

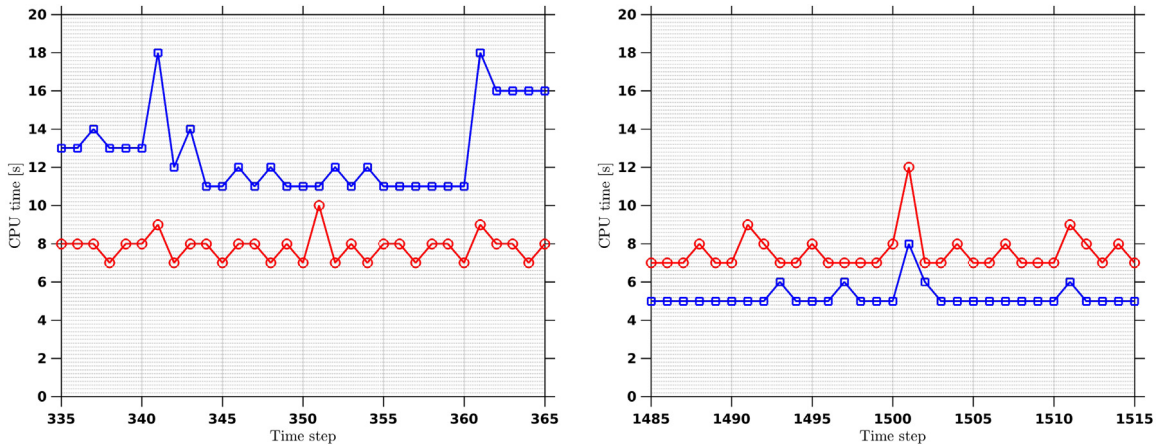


Fig. 14. Comparison of the CPU time for 30 time steps near the peak systole (left) and mid diastole (right). The curves represent the CPU time spent in each time step for the proposed method (red) and the `svSolver` (blue). During systole, the proposed solution method requires 3 Newton–Raphson iterations, while the `svSolver` requires 14 iterations. During diastole, the proposed solution method converges in 2 Newton–Raphson iterations, and the `svSolver` needs 4 iterations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

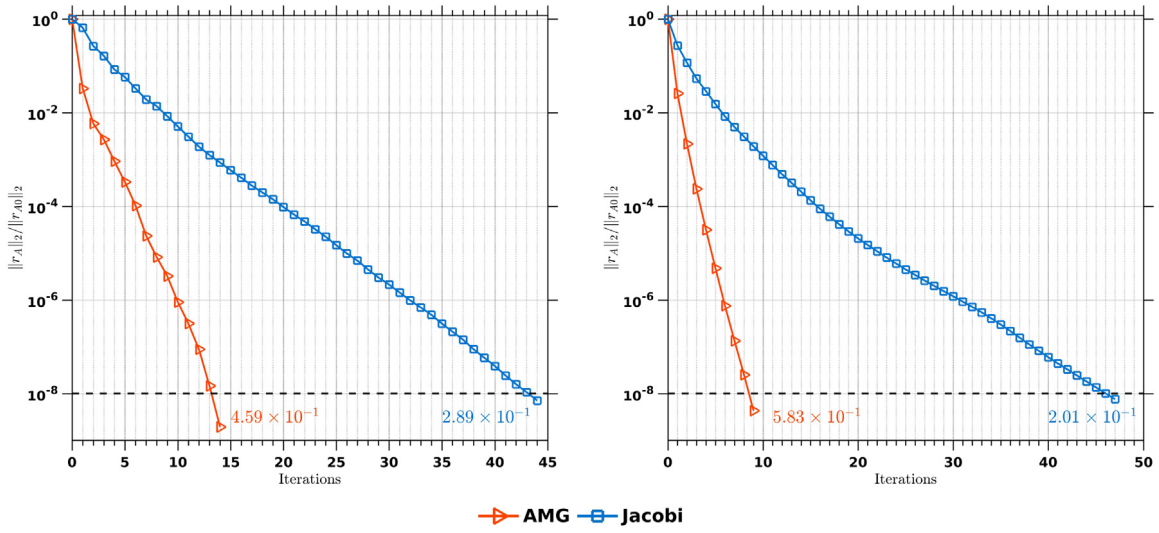


Fig. 15. Convergence history for solving **A** with the AMG and Jacobi preconditioner for the aortofemoral model (left) and the pulmonary model (right). The numbers indicate the CPU time for solving the equations in the unit of seconds.

Table 6
Comparison of the CPU time for the preconditioner setup (T_S) and the total CPU time for solving **A** (T_L) in seconds using the AMG and Jacobi preconditioners. The Courant number is fixed to be 2.0.

n_{eq}	Proc.	AMG		Jacobi	
		T_S (s)	T_L (s)	T_S (s)	T_L (s)
3.10×10^5	6	8.5×10^{-2}	2.3×10^{-1}	3.8×10^{-3}	8.9×10^{-2}
2.48×10^6	48	1.1×10^{-1}	4.3×10^{-1}	5.2×10^{-3}	2.7×10^{-1}
1.98×10^7	384	1.9×10^{-1}	5.3×10^{-1}	8.1×10^{-3}	4.5×10^{-1}

5. Conclusion

In this work, we designed and applied the three-level nested block preconditioning technique to the incompressible Navier–Stokes equations with reduced models coupled on the boundaries. The proposed preconditioning technique combines the merits of the SCR procedure and the conventional physics-based block preconditioners. In particular, it uses an algorithm, rather than an explicit algebraic form, to approximate the behavior of the Schur complement. In comparison with conventional block preconditioners, this approach is well-suited to problems involving multiscale and multiphysics coupling. With a proper setting of the intermediate and inner solvers, the proposed method is demonstrated to attain more robust and more efficient performance than the existing technologies, especially for hemodynamic simulations. One attribute of the nested block preconditioner is the proliferation of options for the intermediate and inner solvers. The choices of the preconditioners and stopping criteria at the intermediate and inner levels critically determine the actual overall performance of the algorithm. For a specific problem, it is often advisable for the practitioners to determine the sweet spot of options to balance robustness and efficiency. In our study, we noticed that using the light-weight Jacobi preconditioner for $\mathbf{A}$ always leads to highly efficient performances. This is somewhat in contrast to the common practices in the traditional block preconditioner, where the AMG preconditioner is recommended for $\mathbf{A}$. As for future work, the preconditioner for $\mathbf{A}$ needs to be further explored and compared for performance optimization purposes. Although this study is based on the variational multiscale formulation, we have applied the same preconditioning technique for the inf–sup stable formulation, and it works well for lower-order element pairs, such as Q_2/Q_1. For higher-order elements, it is tempting to apply this methodology within a Newton–Krylov framework [68–70]. There is still room to further improve the performance of the AMG preconditioner for the Schur complement, particularly in the design of the smoother [60,62,88]. In the last, a crucial next step is to apply the iterative solution method for FSI problems within the unified continuum modeling framework [42,44].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank A.W. Bergersen for helpful discussions on the FDA idealized medical device benchmark. This work is supported by the National Institutes of Health under the award numbers 1R01HL121754, 1R01HL123689, R01EB01830204, the Department of Defense award W81XWH1810518, the computational resources from the Stanford Research Computing Center, and the Extreme Science and Engineering Discovery Environment (XSEDE) supported by the NSF grant ACI-1053575.

References

[1] G. Bao, Y. Bazilevs, J. Chung, P. Decuzzi, H. Espinosa, M. Ferrari, H. Gao, S. Hossain, T. Hughes, R. Kamm, W. Liu, A. Marsden, B. Schrefler, USNCTAM perspectives on mechanics in medicine, J. R. Soc. Interface 11 (97) (2014).
[2] C. Figueroa, C. Taylor, A. Marsden, Blood flow, in: Encyclopedia of Computational Mechanics, 2017, pp. 1–31.
[3] TOP500 lists, https://www.top500.org/.
[4] C. Taylor, T. Fonte, J. Min, Computational fluid dynamics applied to cardiac computed tomography for noninvasive quantification of fractional flow reserve: scientific basis, J. Am. Coll. Cardiol. 61 (2013) 2233–2241.
[5] C. Zarins, C. Taylor, J. Min, Computed fractional flow reserve (FFT CT) derived from coronary CT angiography, J. Cardiovasc. Transl. Res. 6 (2013) 708–714.
[6] F. Hwang, X. Cai, A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier–Stokes equations, J. Comput. Phys. 204 (2005) 666–691.
[7] F. Kong, X. Cai, A scalable nonlinear fluid-structure interaction solver based on a Schwarz preconditioner with isogeometric unstructured coarse spaces in 3D, J. Comput. Phys. 340 (2017) 498–518.
[8] F. Kong, V. Kheyfets, E. Finol, X. Cai, Simulation of unsteady blood flows in a patient-specific compliant pulmonary artery with a highly parallel monolithically coupled fluid-structure interaction algorithm, Int. J. Numer. Methods Biomed. Eng. 35 (2019) e3208.
[9] Y. Wu, X. Cai, A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation, J. Comput. Phys. 258 (2014) 524–537.
[10] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations, J. Comput. Phys. 227 (2008) 1790–1808.

[11] E. Cyr, J. Shadid, R. Tuminaro, Stabilization and scalable block preconditioning for the Navier–Stokes equations, J. Comput. Phys. 231 (2012) 345–363.
[12] T. Washio, T. Hisada, H. Watanabe, T. Tezduyar, A robust preconditioner for fluid-structure interaction problems, Comput. Methods Appl. Mech. Engrg. 194 (2005) 4027–4047.
[13] R. Tuminaro, C. Tong, J. Shadid, K. Devine, D. Day, On a multilevel preconditioning module for unstructured mesh Krylov solvers: two-level Schwarz, Commun. Numer. Methods. Eng. 18 (2002) 383–389.
[14] J. Shadid, R. Tuminaro, K. Devine, G. Hennigan, P. Lin, Performance of fully coupled domain decomposition preconditioners for finite element transport/reaction simulations, J. Comput. Phys. 205 (2005) 24–47.
[15] M. Benzi, G. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137.
[16] A. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comp. 22 (1968) 745–762.
[17] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[18] P. Gresho, R. Sani, Incompressible Flow and the Finite Element Method, in: Advection-Diffusion and Isothermal Laminar Flow, vol. 1, John Wiley and Sons, Inc., New York, NY (United States), 1998.
[19] K. Jansen, C. Whiting, G. Hulbert, A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method, Comput. Methods Appl. Mech. Engrg. 190 (2000) 305–319.
[20] J. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Engrg. 195 (2006) 6011–6045.
[21] H. Elman, V. Howle, J. Shadid, R. Tuminaro, A parallel block multi-level preconditioner for the 3D incompressible Navier–Stokes equations, J. Comput. Phys. 187 (2003) 504–523.
[22] S. Deparis, G. Grandperrin, A. Quarteroni, Parallel preconditioners for the unsteady Navier–Stokes equations and applications to hemodynamics simulations, Comput. & Fluids 92 (2014) 253–273.
[23] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, Block preconditioners based on approximate commutators, SIAM J. Sci. Comput. 27 (2006) 1651–1668.
[24] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state Navier–Stokes equations, SIAM J. Sci. Comput. 24 (2002) 237–256.
[25] J. Shadid, P. Pawlowski, E. Cyr, R. Tuminaro, L. Chacon, P. Weber, Scalable implicit incompressible resistive MHD with stabilized FE and fully-coupled Newton-Krylov-AMG, Comput. Methods Appl. Mech. Engrg. 304 (2016) 1–25.
[26] D. Silvester, H. Elman, D.K.A. Wathen, Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow, J. Comput. Appl. Math. 128 (2001) 261–279.
[27] S. Turek, Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach, Springer Science & Business Media, 1999.
[28] J. Peiró, A. Veneziani, Reduced Models of the Cardiovascular System, Springer, 2009, pp. 347–394,
[29] A. Quarteroni, A. Veneziani, C. Vergara, Geometric multiscale modeling of the cardiovascular system, between theory and practice, Comput. Methods Appl. Mech. Engrg. 302 (2016) 193–252.
[30] C. Figueroa, I. Vignon-Clementel, K. Jansen, T. Hughes, C. Taylor, A coupled momentum method for modeling blood flow in three-dimensional deformable arteries, Comput. Methods Appl. Mech. Engrg. 195 (2006) 5685–5706.
[31] M. Moghadam, Y. Bazilevs, A. Marsden, A new preconditioning technique for implicitly coupled multidomain simulations with applications to hemodynamics, Comput. Mech. 52 (2013) 1141–1152.
[32] M. Moghadam, Y. Bazilevs, A. Marsden, A bi-partitioned iterative algorithm for solving linear systems obtained from incompressible flow problems, Comput. Methods Appl. Mech. Engrg. 286 (2015) 40–62.
[33] SimVascular simulation guide, 2019, http://simvascular.github.io/docsFlowSolver.html. (Accessed: 2019-08-05).
[34] A. Updegrove, N. Wilson, J. Merkow, H. Lan, A. Marsden, S. Shadden, SimVascular: An open source pipeline for cardiovascular simulation, Ann. Biomed. Eng. 45 (2017) 525–541.
[35] D. May, L. Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, Phys. Earch Planet. Inter. 171 (2008) 33–47.
[36] E. Cyr, J. Shadid, R. Tuminaro, Teko: A block preconditioning capability with concrete example applications in Navier–Stokes and MHD, SIAM J. Sci. Comput. 38 (2016) S307–S331.
[37] M. Manguoglu, A. Sameh, T. Tezduyar, S. Sathe, A nested iterative scheme for computation of incompressible flows in long domains, Comput. Mech. 43 (2008) 73–80.
[38] M. Manguoglu, K. Takizawa, A. Sameh, T. Tezduyar, Nested and parallel sparse algorithms for arterial fluid mechanics computations with boundary layer mesh refinement, Internat. J. Numer. Methods Fluids 65 (2011) 135–149.
[39] M. Manguoglu, A. Sameh, F. Saied, T. Tezduyar, S. Sathe, Preconditioning techniques for nonsymmetric linear systems in the computation of incompressible flows, J. Appl. Mech. 76 (2009) 021204.
[40] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (1993) 461–469.
[41] J. Liu, A. Marsden, A robust iterative method for finite elastodynamics with nested block preconditioning, J. Comput. Phys. 383 (2019) 72–93.
[42] J. Liu, A. Marsden, A unified continuum and variational multiscale formulation for fluids, solids, and fluid-structure interaction, Comput. Methods Appl. Mech. Engrg. 337 (2018) 549–597.
[43] J. Liu, A. Marsden, Z. Tao, An energy-stable mixed formulation for isogeometric analysis of incompressible hyper-elastodynamics, Internat. J. Numer. Methods Engrg. 120 (2019) 937–963.
[44] J. Liu, W. Yang, I.S. Lan, A.L. Marsden, Fluid-structure interaction modeling of blood flow in the pulmonary arteries using the unified continuum and variational multiscale formulation, Mech. Res. Commun. (2020) accepted for publication.

[45] C. Taylor, T. Hughes, C. Zarins, Finite element modeling of blood flow in arteries, Comput. Methods Appl. Mech. Engrg. 158 (1998) 155–196.
[46] Y. Bazilevs, V. Calo, J. Cottrell, T. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Comput. Methods Appl. Mech. Engrg. 197 (2007) 173–201.
[47] J. Liu, I. Lan, O. Tikenogullari, A. Marsden, A note on the accuracy of the generalized-α scheme for the incompressible Navier–Stokes equations, 2020, arXiv preprint arXiv:2002.01098.
[48] M. Moghadam, I. Vignon-Clementel, R. Figliola, A. Marsden, MOCHA, A modular numerical method for implicit 0D/3D coupling in cardiovascular finite element simulations, J. Comput. Phys. 244 (2013) 63–79.
[49] L. Pauli, M. Behr, On stabilized space-time FEM for anisotropic meshes: Incompressible Navier–Stokes equations and applications to blood flow in medical devices, Internat. J. Numer. Methods Fluids 85 (2017) 189–209.
[50] M. von Danwitz, V. Karyofylli, N. Hosters, M. Behr, Simplex space-time meshes in compressible flow simulations, Internat. J. Numer. Methods Fluids 91 (2019) 29–48.
[51] K. Takizawa, J. Christopher, T. Tezduyar, S. Sathe, Space-time finite element computation of arterial fluid-structure interactions with patient-specific data, Int. J. Numer. Methods Biomed. Eng. 26 (2010) 101–116.
[52] L. Franca, S. Frey, Stabilized finite element methods: II. The incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 99 (1992) 209–233.
[53] C. Figueroa, A Coupled-Momentum Method to Model Blood Flow and Vessel Deformation in Human Arteries: Applications in Disease Research and Simulation-Based Medical Planning (Ph.D. thesis), Stanford university, 2006.
[54] Y. Bazilevs, J. Gohean, T. Hughes, R. Moser, Y. Zhang, Patient-specific isogeometric fluid–structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device, Comput. Methods Appl. Mech. Engrg. 198 (2009) 3534–3550.
[55] C. Bertoglio, A. Caiazzo, A Stokes-residual backflow stabilization method applied to physiological flows, J. Comput. Phys. 313 (2016) 260–278.
[56] M. Moghadam, Y. Bazilevs, T. Hsia, I. Vignon-Clementel, A. Marsden, A comparison of outlet boundary treatments for prevention of backflow divergence with relevance to blood flow simulation, Comput. Mech. 48 (2011) 277–291.
[57] C. Bertoglio, A. Caiazzo, Y. Bazilevs, M. Braack, M. Esmaily, V. Gravemeier, A. Marsden, O. Pironneau, I. Vignon-Clementel, W. Wall, Benchmark problems for numerical treatment of backflow at open boundaries, Int. J. Numer. Methods Biomed. Eng. 34 (2018) e2918.
[58] J. Chung, G. Hulbert, A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method, J. Appl. Mech. 60 (1993) 371–375.
[59] Y. Saad, M. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[60] R. Falgout, U. Yang, Hypre: A library of high performance preconditioners, in: International Conference on Computational Science, Springer, 2002, pp. 632–641.
[61] Hypre/BoomerAMG preconditioning, 2019, https://mooseframework.org/application_development/hypre.html. (Accessed: 2019-08-19).
[62] A. Baker, R. Falgout, T. Kolev, U. Yang, Multigrid smoothers for ultraparallel computing, SIAM J. Sci. Comput. 33 (2011) 2864–2887.
[63] H. Elman, D. Silvester, A. Wathen, Block preconditioners for the discrete incompressible Navier–Stokes equations, Internat. J. Numer. Methods Fluids 40 (2002) 333–344.
[64] I. Ipsen, A note on preconditioning nonsymmetric matrices, SIAM J. Sci. Comput. 23 (2001) 1050–1051.
[65] M. Murphy, G. Golub, A. Wathen, A note on preconditioning for indefinite linear systems, SIAM J. Sci. Comput. 21 (2000) 1969–1972.
[66] R. Anderson, A. Barker, J. Bramwell, J. Cerveny, J. Dahm, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, M. Stowell, V. Tomov, MFEM: A modular finite element methods library, 2018.
[67] D. Knoll, D. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[68] J. Brown, Efficient nonlinear solvers for nodal high-order finite elements in 3D, J. Sci. Comput. 45 (2010) 48–63.
[69] M. Pernice, M. Tocci, A multigrid-preconditioned Newton–Krylov method for the Incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 23 (2001) 398–418.
[70] D. Davydov, J. Pelteret, D. Arndt, P. Steinmann, A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid, Internat. J. Numer. Methods Engrg. (2020).
[71] S. Patankar, D. Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, in: Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion, Elsevier, 1983, pp. 54–73.
[72] S. Deparis, D. Forti, G. Grandperrin, A. Quarteroni, FaSCI: A block parallel preconditioner for fluid–structure interaction in hemodynamics, J. Comput. Phys. 327 (2016) 700–718.
[73] J. White, R. Borja, Block-preconditioned Newton–Krylov solvers for fully coupled flow and geomechanics, Comput. Geosci. 15 (2011) 647.
[74] F. Verdugo, W. Wall, Unified computational framework for the efficient solution of n-field coupled problems with monolithic schemes, Comput. Methods Appl. Mech. Engrg. 310 (2016) 335–366.
[75] A. Quarteroni, F. Saleri, A. Veneziani, Factorization methods for the numerical approximation of Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 188 (2000).
[76] M. Akgün, J. Garcelon, R. Haftka, Fast exact linear and non-linear structural reanalysis and the Sherman–Morrison–Woodbury formulas, Internat. J. Numer. Methods Engrg. 50 (2001) 1587–1606.
[77] E. Phillips, E. Cyr, J. Shadid, An investigation of block preconditioners for unsteady Navier–Stokes, in: CSRI SUMMER PROCEEDINGS, 2010.

[78] Stampede2 user guide, 2019, https://portal.tacc.utexas.edu/user-guides/stampede2. (Accessed: 2019-06-23).
[79] A. Bergersen, M. Mortensen, K. Valen-Sendstad, The FDA nozzle benchmark: "In theory there is no difference between theory and practice, but in practice there is", Int. J. Numer. Methods Biomed. Eng. 35 (2019) e3150.
[80] C. Geuzaine, J. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, Int. J. Numer. Methods Biomed. Eng. 79 (2009) 1309–1331.
[81] V. Zmijanovic, S. Mendez, V. Moureau, F. Nicoud, About the numerical robustness of biomedical benchmark cases: interlaboratory FDA's idealized medical device, Int. J. Numer. Methods Biomed. Eng. 33 (2017) e02789.
[82] The FDA's "critical path" computational fluid dynamics (CFD)/blood damage project, 2019, https://nciphub.org/wiki/FDA_CFD. (Accessed: 2019-03-28).
[83] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, D. May, L. McInnes, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, PETSC Users Manual, Tech. Rep. ANL-95/11 - Revision 3.8, Argonne National Laboratory, 2017.
[84] H. Si, TetGen, a Delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Software 41 (2015) 11.
[85] Cardiovascular and pulmonary model repository, http://www.vascularmodel.com.
[86] Simmetrix, http://www.simmetrix.com/.
[87] C. Burstedde, O. Ghattas, G. Stadler, T. Tu, L. Wilcox, Parallel scalable adjoint-based adaptive solution of variable-viscosity Stokes flow problems, Comput. Methods Appl. Mech. Engrg. 198 (2009) 1691–1700.
[88] P. Lin, J. Shadid, P. Tsuji, On the performance of Krylov smoothing for fully coupled AMG preconditioners for VMS resistive MHD, Internat. J. Numer. Methods Engrg. 120 (12) (2019) 1297–1309.