Journal of Computational Physics 383 (2019) 72–93

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

# A robust and efficient iterative method for hyper-elastodynamics with nested block preconditioning

Ju Liu *, Alison L. Marsden

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

ARTICLE INFO

*Article history:*
Received 12 June 2018
Received in revised form 14 January 2019
Accepted 18 January 2019
Available online 1 February 2019

*Keywords:*
Variational multiscale method
Saddle-point problem
Nested iterative method
Block preconditioner
Anisotropic incompressible hyperelasticity
Arterial wall model

ABSTRACT

We develop a robust and efficient iterative method for hyper-elastodynamics based on a novel continuum formulation recently developed in [1]. The numerical scheme is constructed based on the variational multiscale formulation and the generalized-$\alpha$ method. Within the nonlinear solution procedure, a block factorization is performed for the consistent tangent matrix to decouple the kinematics from the balance laws. Within the linear solution procedure, another block factorization is performed to decouple the mass balance equation from the linear momentum balance equations. A nested block preconditioning technique is proposed to combine the Schur complement reduction approach with the fully coupled approach. This preconditioning technique, together with the Krylov subspace method, constitutes a novel iterative method for solving hyper-elastodynamics. We demonstrate the efficacy of the proposed preconditioning technique by comparing with the SIMPLE preconditioner and the one-level domain decomposition preconditioner. Two representative examples are studied: the compression of an isotropic hyperelastic cube and the tensile test of a fully-incompressible anisotropic hyperelastic arterial wall model. The robustness with respect to material properties and the parallel performance of the preconditioner are examined.



## 1. Introduction

In our recent work [1], a unified continuum modeling framework was developed. In this framework, hyperelastic solids and viscous fluids are distinguished only through the deviatoric part of the Cauchy stress, in contrast to prior modeling approaches. In our derivation, the Gibbs free energy, rather than the Helmholtz free energy, is chosen as the thermodynamic potential, resulting in a unified model for compressible and incompressible materials. A beneficial outcome of the modeling framework is that it naturally allows one to apply a computational fluid dynamics (CFD) algorithm to solid dynamics, or vice versa. In our work [1], the variational multiscale (VMS) analysis, a mature numerical modeling approach in CFD [2], is taken to design the spatial discretization for solid dynamics. This numerical model provides a stabilization mechanism that circumvents the Ladyzhenskaya–Babuška–Brezzi (LBB) condition for equal-order interpolations. In particular, it allows one to use low-order tetrahedral elements, even for fully incompressible materials. This gives us the maximum flexibility in geometrical modeling and mesh generation.

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

https://doi.org/10.1016/j.jcp.2019.01.019
0021-9991/

In this work, we build upon the proposed unified formulation to develop a robust and efficient iterative method. Traditional black-box preconditioners are non-robust, and the convergence rate of the linear solver drops significantly under certain conditions. The lack of robustness may be attributed to the saddle-point nature of the problem. Algebraic preconditioners built based on incomplete factorizations are prone to fail due to zero-pivoting; one-level domain decomposition preconditioners do not perform well due to its locality. In this work, we design a preconditioning technique tailored for the VMS formulation for hyper-elastodynamics [1]. The design of the preconditioner is based on a nested block factorization of the consistent tangent matrix in the Newton–Raphson iteration. A block factorization is performed in the nonlinear solution procedure to decouple the kinematics from the balance laws [3]. The resulting $2 \times 2$ block matrix is further factorized in the linear solution procedure. This strategy is, in part, related to the classical projection method [4,5] and the block preconditioning technique [6–8] that have been widely used in the CFD community. We examine the solver performance for both isotropic and anisotropic hyperelastic models. The significance of this work is that it paves the way towards robust, efficient, and scalable implicit solver technology for biomechanics and monolithic fluid–solid interaction (FSI) simulations [1]. In the rest part of this section, we give an overview of the background and an outline of the work.

### 1.1. Projection method and block preconditioners

The development of efficient solver techniques for multiphysics problems has been an active area of research in recent years [9]. One simple but important prototype multiphysics problem is the Stokes or the Navier–Stokes equations, representing the coupling between the mass conservation and the balance of the linear momentum for incompressible flows. In the late 1960s, the Chorin–Teman projection method [4,5] was proposed to solve for the pressure and the velocity separately based on the Helmholtz decomposition. Since then, the projection method and its variants have attracted concentrated research and lead to a voluminous literature [10–13]. The projection method is attractive because the nonlinear system of equations is decomposed into a series of linear elliptic equations. Although this method has attracted significant attention, it still poses several major challenges. One critical issue is that the physics-based splitting necessitates the introduction of an artificial boundary condition for the pressure. There is no general theory to guide the choice of the artificial boundary conditions, and most likely this artificial boundary condition limits the solution accuracy. For an overview of the projection method, the readers are referred to the review article [14].

In recent years, it has been realized that one can invoke an arbitrary time stepping scheme (e.g. fully implicit) and achieve the decoupling of physics within the linear solver. Indeed, in each iteration of the Krylov subspace method, one only needs to solve with a preconditioner and perform a matrix–vector multiplication to construct the new search direction. Therefore, if the preconditioner is endowed with a block structure, one may sequentially solve each block matrix with less cost. It has been pointed out that the Chorin–Teman projection method is closely related to a block preconditioner [15]. Consider a matrix problem with a $2 \times 2$ block structure,

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

This matrix can be factored into lower triangular, diagonal, and upper triangular matrices as follows,

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

wherein $\mathbf{I}$ is the identity matrix, and $\mathbf{O}$ is the zero matrix. The diagonal block matrix $\mathcal{D}$ contains a Schur complement $\mathbf{S} := \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$, which acts as an algebraic analogue of the Laplacian operator for the pressure field [16]. To construct a preconditioner for $\mathcal{A}$, one needs to provide approximations for $\mathbf{A}$ and $\mathbf{S}$ that can be conveniently solved with. The new formulation for hyper-elastodynamics we consider here is similar to the generalized Stokes equations, in which the operator $\mathbf{A}$ arises from the discretization of a combination of zeroth order and second order differential operators. Thus, $\mathbf{A}$ is amenable for approximation by a standard preconditioning technique. Due to the presence of $\mathbf{A}^{-1}$, $\mathbf{S}$ is a dense matrix. When the matrix $\mathbf{A}$ represents a discretization of a zeroth-order differential operator, an effective choice is to replace $\mathbf{S}$ by $\hat{\mathbf{S}} := \mathbf{D} - \mathbf{C}\left(\mathrm{diag}\left(\mathbf{A}\right)\right)^{-1}\mathbf{B}$ to construct the preconditioner for $\mathcal{A}$. This choice is closely related to the SIMPLE scheme commonly used in CFD [17,18]. When the matrix $\mathbf{A}$ represents a discretization of a second-order differential operator, a scaled mass matrix is often effective [19]. For more complicated problems, designing a spectrally equivalent preconditioner for the Schur complement is challenging and, in a broad sense, remains an open question. In recent years, progress has been made for problems where $\mathbf{A}$ is dominated by a discrete convection operator. Notable examples include the BFBT preconditioner [20], the pressure convection diffusion preconditioner [21], and the least squares commutator (LSC) preconditioner [22]. Based on the Sherman–Morrison formula, a different preconditioner for the Schur complement can be designed for problems with significant contributions from the boundary conditions [23]. In all, the block preconditioner, as an algebraic interpretation of the projection method, has become increasingly popular, since it does not necessitate ad hoc pressure boundary conditions and allows fully implicit time stepping schemes.

If one can solve the sub-matrices $\mathbf{A}$ and $\mathbf{S}$ to a prescribed tolerance, the matrix $\mathcal{A}$ is solved in one pass without generating a Krylov subspace. This is commonly known as the *Schur complement reduction* (SCR) or *segregated* approach [6,24–26]. In

contrast, the aforementioned strategy, where $\mathcal{A}$ is solved by a preconditioned iterative method, is referred to as the *coupled* approach [6]. For many problems, it is impractical to explicitly construct the Schur complement. Still, the action of the Schur complement on a vector can be obtained in a "matrix-free" manner (see Algorithm 2 in Section 4.2). Thus, one can still solve with the Schur complement by iterative methods. To achieve high accuracy, a sufficient number of bases of the Krylov subspace for $\mathbf{S}$ need to be generated, and this procedure can be prohibitively expensive.

### 1.2. Nested preconditioning technique

The difference between the coupled approach and the segregated approach can be viewed as follows. In the coupled approach, $\mathbf{S}$ is replaced by a sparse approximation to generate a preconditioner for $\mathcal{A}$. In the segregated approach or SCR, one strives to solve directly with $\mathbf{S}$. The distinction between the two approaches is blurred by using the SCR procedure as a preconditioner. In doing so, one does not need to solve with $\mathbf{S}$ to a high precision, thus alleviating the computational burden. In comparison with the coupled approach, the information of the Schur complement is maintained in the preconditioner (up to the tolerances of SCR), and this will improve the robustness. Therefore, in solving with $\mathcal{A}$, there are three nested levels. In the outer level, a Krylov subspace method is applied for $\mathcal{A}$ with a block preconditioner. In the intermediate level, the block preconditioner is applied by solving with the matrices $\mathbf{A}$ and $\mathbf{S}$. In the inner level, a solver of $\mathbf{A}$ is invoked to approximate the action of $\mathbf{S}$ on a vector. Two mechanisms guarantee and accelerate the convergence. In the outer level, the Krylov subspace method for $\mathcal{A}$ minimizes the residual of the coupled problem. In the intermediate and inner levels, the SCR procedure is utilized as the preconditioner, which itself can be viewed as an inaccurate solver for $\mathcal{A}$.

Using SCR as a preconditioner was first proposed within a Richardson iteration scheme [27]. Due to the symmetry property of that problem, a conjugate gradient method is applied to solve the Schur complement equation. Later, the nested iterative scheme was investigated for CFD problems [28,29], and the reported results indicate that using the SCR procedure as a preconditioner in a Richardson iteration outperforms the coupled approach with a Krylov subspace method. The nested algorithm was then further investigated using the biconjugate gradient stabilized method (BiCGStab) as the outer solver [30]. The nested iterative scheme in [30] uses rather crude stopping criteria for the intermediate and inner solvers. Still, its performance is superior to that of BiCGStab preconditioned by a BFBT preconditioner.

Our investigation of the VMS formulation for hyper-elastodynamics starts with a SIMPLE-type block preconditioner using our in-house code [31]. As will be shown in Section 5, the Krylov subspace method with a block preconditioner like SIMPLE is not always robust. This can be attributed to the ignorance of the off-diagonal entries in $\mathbf{A}$. Because of that, it is appealing to consider preconditioners like LSC, since the off-diagonal information of $\mathbf{A}$ is maintained. However, non-convergence has been reported for LSC when solving the Navier-Stokes equations with stabilized finite element schemes [32]. We then ruled out this option since our VMS formulation involves a similar pressure stabilization term. Consequently, we consider using SCR with relaxed tolerances as a preconditioner. In doing so, the Schur complement is approximated through using an inner solver. In contrast to the nested iterative approaches introduced above, we adopt the following techniques in our study: (1) we use GMRES [33] and its variant [34] as the Krylov subspace method in all three levels to leverage their robustness in handling non-symmetric matrix problems; (2) we apply the algebraic multigrid (AMG) preconditioner [35] for problems at the intermediate level to enhance the robustness of the overall algorithm; (3) we use the sparse approximation $\hat{\mathbf{S}}$ as a preconditioner when solving with $\mathbf{S}$. We demonstrate application of this method to hyper-elastodynamics, however we anticipate its general use in CFD and FSI problems in future work.

### 1.3. Structure and content of the paper

The remainder of the study is organized as follows. In Section 2, we state the governing equations of hyper-elastodynamics [1]. In Section 3, the numerical scheme is presented. A block factorization for the consistent tangent matrix is performed to reduce the size of the linear algebra problem. In Section 4, the nested block preconditioning technique is discussed in detail. In Section 5, we present two representative examples to demonstrate the efficacy of the proposed solver technology. The first example is the compression of an isotropic elastic cube [36], and the second is the tensile test of a fully incompressible anisotropic hyperelastic arterial wall model [37]. Comparisons with other preconditioners are made. We draw conclusions in Section 6.

## 2. Hyper-elastodynamics

In this section, we state the initial–boundary value problem for hyper-elastodynamics, following the derivation in [1]. Let $\Omega_{\boldsymbol{X}}$ and $\Omega_{\boldsymbol{x}}$ be bounded open sets in $\mathbb{R}^{n_{sd}}$ with Lipschitz boundaries, where $n_{sd}$ represents the number of space dimensions. They represent the initial and the current configurations of the body, respectively. The motion of the body is described by a family of diffeomorphisms, parametrized by the time coordinate $t$,

$$\begin{aligned}\boldsymbol{\varphi}_t(\cdot)=\boldsymbol{\varphi}(\cdot,t):\Omega_{\boldsymbol{X}}\to\Omega_{\boldsymbol{x}}=\boldsymbol{\varphi}(\Omega_{\boldsymbol{X}},t)=\boldsymbol{\varphi}_t(\Omega_{\boldsymbol{X}}),\quad \forall t\geq 0,\\ \boldsymbol{X}\mapsto\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)=\boldsymbol{\varphi}_t(\boldsymbol{X}),\quad \forall\boldsymbol{X}\in\Omega_{\boldsymbol{X}}.\end{aligned}$$

In the above, $\boldsymbol{x}$ is the current position of a material particle originally located at $\boldsymbol{X}$. This requires that $\boldsymbol{\varphi}(\boldsymbol{X},0)=\boldsymbol{X}$. The displacement and velocity of the material particle are defined as

$$\boldsymbol{U}:=\boldsymbol{\varphi}(\boldsymbol{X},t)-\boldsymbol{\varphi}(\boldsymbol{X},0)=\boldsymbol{\varphi}(\boldsymbol{X},t)-\boldsymbol{X},\quad \boldsymbol{V}:=\left.\frac{\partial\boldsymbol{\varphi}}{\partial t}\right|_{\boldsymbol{X}}=\left.\frac{\partial\boldsymbol{U}}{\partial t}\right|_{\boldsymbol{X}}=\frac{d\boldsymbol{U}}{dt}.$$

In the definition of $\boldsymbol{V}$ and in what follows, $d(\cdot)/dt$ designates a total time derivative. The spatial velocity is defined as $\boldsymbol{v}:=\boldsymbol{V}\circ\boldsymbol{\varphi}_t^{-1}$. Analogously, we define $\boldsymbol{u}:=\boldsymbol{U}\circ\varphi_t^{-1}$. The deformation gradient, the Jacobian determinant, and the right Cauchy–Green tensor are defined as

$$\boldsymbol{F}:=\frac{\partial\boldsymbol{\varphi}}{\partial\boldsymbol{X}},\qquad J:=\det(\boldsymbol{F}),\qquad \boldsymbol{C}:=\boldsymbol{F}^T\boldsymbol{F}.$$

We define $\tilde{\boldsymbol{F}}$ and $\tilde{\boldsymbol{C}}$ as

$$\tilde{\boldsymbol{F}}:=J^{-\frac{1}{3}}\boldsymbol{F},\qquad \tilde{\boldsymbol{C}}:=J^{-\frac{2}{3}}\boldsymbol{C},$$

which represent the distortional parts of $\boldsymbol{F}$ and $\boldsymbol{C}$, respectively. We denote the thermodynamic pressure of the continuum body as $p$. The mechanical behavior of an elastic material can be described by a Gibbs free energy $G(\tilde{\boldsymbol{C}},p)$. In [1], it is shown that the Gibbs free energy enjoys a decoupled structure,

$$G(\tilde{\boldsymbol{C}},p)=G_{ich}(\tilde{\boldsymbol{C}})+G_{vol}(p),$$

where $G_{ich}$ and $G_{vol}$ represent the isochoric and volumetric elastic responses. Under the isothermal condition, the energy equation is decoupled from the system, and it suffices to consider the following equations for the motion of the continuum body,

$$\boldsymbol{0}=\frac{d\boldsymbol{u}}{dt}-\boldsymbol{v},\qquad \text{in } \Omega_{\boldsymbol{x}}, \tag{2.1}$$

$$\boldsymbol{0}=\rho(p)\frac{d\boldsymbol{v}}{dt}-\nabla_{\boldsymbol{x}}\cdot\boldsymbol{\sigma}^{dev}+\nabla_{\boldsymbol{x}}p-\rho(p)\boldsymbol{b},\qquad \text{in } \Omega_{\boldsymbol{x}}, \tag{2.2}$$

$$0=\beta(p)\frac{dp}{dt}+\nabla_{\boldsymbol{x}}\cdot\boldsymbol{v}\qquad \text{in } \Omega_{\boldsymbol{x}}. \tag{2.3}$$

In the above, $\beta(p)$ is the isothermal compressibility coefficient, $\rho(p)$ denotes the density in the current configuration, and $\boldsymbol{\sigma}^{dev}$ represents the deviatoric part of the Cauchy stress. Equations (2.1) describe the kinematic relation between the displacement and the velocity, and equations (2.2) and (2.3) describe the balance of linear momentum and mass. The constitutive relations of the elastic material are represented in terms of the Gibbs free energy as follows,

$$\rho(p):=\left(\frac{dG_{vol}}{dp}\right)^{-1},\quad \beta_\theta(p):=\frac{1}{\rho}\frac{d\rho}{dp}=-\frac{\partial^2G_{vol}}{\partial p^2}\Big/\frac{\partial G_{vol}}{\partial p},\quad \boldsymbol{\sigma}^{dev}:=J^{-1}\tilde{\boldsymbol{F}}\left(\mathbb{P}:\tilde{\boldsymbol{S}}\right)\tilde{\boldsymbol{F}}^T, \tag{2.4}$$

wherein the projector $\mathbb{P}$ and the fictitious second Piola–Kirchhoff stress $\tilde{\boldsymbol{S}}$ are defined as

$$\mathbb{P}:=\mathbb{I}-\frac{1}{3}\boldsymbol{C}^{-1}\otimes\boldsymbol{C},\quad \tilde{\boldsymbol{S}}:=2\frac{\partial(\rho_0G)}{\partial\tilde{\boldsymbol{C}}}=2\frac{\partial(\rho_0G_{ich})}{\partial\tilde{\boldsymbol{C}}},$$

$\mathbb{I}$ is the fourth-order identity tensor, and $\rho_0$ is the density in the referential configuration. Interested readers are referred to [1] for a detailed derivation of the governing equations and the constitutive relations. The boundary $\Gamma_{\boldsymbol{x}}=\partial\Omega_{\boldsymbol{x}}$ can be partitioned into two non-overlapping subdivisions: $\Gamma_{\boldsymbol{x}}=\Gamma_{\boldsymbol{x}}^g\cup\Gamma_{\boldsymbol{x}}^h$, wherein $\Gamma_{\boldsymbol{x}}^g$ is the Dirichlet part of the boundary, and $\Gamma_{\boldsymbol{x}}^h$ is the Neumann part of the boundary. Boundary conditions can be stated as

$$\boldsymbol{u}=\boldsymbol{g}\quad \text{on } \Gamma_{\boldsymbol{x}}^g,\quad \boldsymbol{v}=\frac{d\boldsymbol{g}}{dt}\quad \text{on } \Gamma_{\boldsymbol{x}}^g,\quad (\boldsymbol{\sigma}^{dev}-p\boldsymbol{I})\boldsymbol{n}=\boldsymbol{h}\quad \text{on } \Gamma_{\boldsymbol{x}}^h. \tag{2.5}$$

Given the initial data $\boldsymbol{u}_0$, $\boldsymbol{v}_0$, and $p_0$, the initial conditions can be stated as

$$\boldsymbol{u}(\boldsymbol{x},0)=\boldsymbol{u}_0(\boldsymbol{x}),\quad \boldsymbol{v}(\boldsymbol{x},0)=\boldsymbol{v}_0(\boldsymbol{x}),\quad p(\boldsymbol{x},0)=p_0(\boldsymbol{x}). \tag{2.6}$$

The equations (2.1)–(2.6) constitute an initial–boundary value problem for hyper-elastodynamics.

## 3. Numerical formulation

In this section, we present the numerical formulation for the strong-form problem. The spatial discretization is based on a VMS formulation [1,2], and the temporal scheme is based on the generalized-$\alpha$ scheme [1,38]. A block factorization, originally introduced in [3], is performed to consistently reduce the size of the linear algebra problem in the Newton–Raphson iterative algorithm.

### 3.1. Variational multiscale formulation

We consider a partition of $\bar{\Omega}_{\boldsymbol{x}}$ by $n_{el}$ non-overlapping, shape-regular elements $\Omega^e_{\boldsymbol{x}}$. The diameter of an element $\Omega^e_{\boldsymbol{x}}$ is denoted by $h^e$, and the maximum diameter of the elements is denoted as $h$. Let $\mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right)$ denote the space of complete polynomials of order $\mathsf{k}$ on $\bar{\Omega}^e_{\boldsymbol{x}}$. The finite element trial solution spaces for the displacement, velocity, and pressure are defined as

$$\mathcal{S}_{\boldsymbol{u}_h} = \left\{ \boldsymbol{u}_h \mid \boldsymbol{u}_h(\cdot,t) \in \left(C^0(\Omega_{\boldsymbol{x}})\right)^{n_{sd}}, t\in[0,T],\ \boldsymbol{u}_h|_{\Omega^e_{\boldsymbol{x}}} \in \left(\mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right)\right)^{n_{sd}},\ \boldsymbol{u}_h(\cdot,t) = \boldsymbol{g} \text{ on } \Gamma^g_{\boldsymbol{x}} \right\},$$
$$\mathcal{S}_{\boldsymbol{v}_h} = \left\{ \boldsymbol{v}_h \mid \boldsymbol{v}_h(\cdot,t) \in \left(C^0(\Omega_{\boldsymbol{x}})\right)^{n_{sd}}, t\in[0,T],\ \boldsymbol{v}_h|_{\Omega^e_{\boldsymbol{x}}} \in \left(\mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right)\right)^{n_{sd}},\ \boldsymbol{v}_h(\cdot,t) = \frac{d\boldsymbol{g}}{dt} \text{ on } \Gamma^g_{\boldsymbol{x}} \right\},$$
$$\mathcal{S}_{p_h} = \left\{ p_h \mid p_h(\cdot,t) \in C^0(\Omega_{\boldsymbol{x}}), t\in[0,T],\ p_h|_{\Omega^e_{\boldsymbol{x}}} \in \mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right) \right\},$$

and the corresponding test function spaces are defined as

$$\mathcal{V}_{\boldsymbol{u}_h} = \left\{ \boldsymbol{w}_{\boldsymbol{u}_h} \mid \boldsymbol{w}_{\boldsymbol{u}_h} \in \left(C^0(\Omega_{\boldsymbol{x}})\right)^{n_{sd}},\ \boldsymbol{w}_{\boldsymbol{u}_h}\big|_{\Omega^e_{\boldsymbol{x}}} \in \left(\mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right)\right)^{n_{sd}},\ \boldsymbol{w}_{\boldsymbol{u}_h} = \boldsymbol{0} \text{ on } \Gamma^g_{\boldsymbol{x}} \right\},$$
$$\mathcal{V}_{\boldsymbol{v}_h} = \left\{ \boldsymbol{w}_{\boldsymbol{v}_h} \mid \boldsymbol{w}_{\boldsymbol{v}_h} \in \left(C^0(\Omega_{\boldsymbol{x}})\right)^{n_{sd}},\ \boldsymbol{w}_{\boldsymbol{v}_h}\big|_{\Omega^e_{\boldsymbol{x}}} \in \left(\mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right)\right)^{n_{sd}},\ \boldsymbol{w}_{\boldsymbol{v}_h} = \boldsymbol{0} \text{ on } \Gamma^g_{\boldsymbol{x}} \right\},$$
$$\mathcal{V}_{p_h} = \left\{ w_{p_h} \mid w_{p_h} \in C^0(\Omega_{\boldsymbol{x}}),\ w_{p_h}\big|_{\Omega^e_{\boldsymbol{x}}} \in \mathsf{P}_{\mathsf{k}}\left(\Omega^e_{\boldsymbol{x}}\right) \right\}.$$

The semi-discrete formulation can be stated as follows. Find $\boldsymbol{y}_h(t) := \{\boldsymbol{u}_h(t), \boldsymbol{v}_h(t), p_h(t)\}^T \in \mathcal{S}_{\boldsymbol{u}_h} \times \mathcal{S}_{\boldsymbol{v}_h} \times \mathcal{S}_{p_h}$ such that for $t\in[0,T]$,

$$0 = \mathbf{B}_k\left(\boldsymbol{w}_{\boldsymbol{u}_h}; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) := \int_{\Omega_{\boldsymbol{x}}} \boldsymbol{w}_{\boldsymbol{u}_h} \cdot \left( \frac{d\boldsymbol{u}_h}{dt} - \boldsymbol{v}_h \right) d\Omega_{\boldsymbol{x}}, \tag{3.1}$$

$$0 = \mathbf{B}_m\left(\boldsymbol{w}_{\boldsymbol{v}_h}; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) := \int_{\Omega_{\boldsymbol{x}}} \boldsymbol{w}_{\boldsymbol{v}_h} \cdot \rho(p_h)\frac{d\boldsymbol{v}_h}{dt} + \nabla_{\boldsymbol{x}}\boldsymbol{w}_{\boldsymbol{v}_h} : \boldsymbol{\sigma}^{dev} - \nabla_{\boldsymbol{x}}\cdot\boldsymbol{w}_{\boldsymbol{v}_h} p_h - \boldsymbol{w}_{\boldsymbol{v}_h}\cdot\rho(p_h)\boldsymbol{b} d\Omega_{\boldsymbol{x}} - \int_{\Gamma^h_{\boldsymbol{x}}} \boldsymbol{w}_{\boldsymbol{v}_h}\cdot\boldsymbol{h} d\Gamma_{\boldsymbol{x}}, \tag{3.2}$$

$$0 = \mathbf{B}_p\left(w_{p_h}; \dot{\boldsymbol{y}}_h, \boldsymbol{y}_h\right) := \int_{\Omega_{\boldsymbol{x}}} w_{p_h}\left(\beta(p_h)\frac{dp_h}{dt} + \nabla_{\boldsymbol{x}}\cdot\boldsymbol{v}_h\right) d\Omega_{\boldsymbol{x}}$$
$$+ \sum_e \int_{\Omega^e_{\boldsymbol{x}}} \boldsymbol{\tau}^e_M \nabla_{\boldsymbol{x}} w_{p_h} \cdot \left( \rho(p_h)\frac{d\boldsymbol{v}_h}{dt} - \nabla_{\boldsymbol{x}}\cdot\boldsymbol{\sigma}^{dev} + \nabla_{\boldsymbol{x}} p_h - \rho(p_h)\boldsymbol{b} \right) d\Omega_{\boldsymbol{x}}, \tag{3.3}$$

for $\forall \left\{\boldsymbol{w}_{\boldsymbol{u}_h}, \boldsymbol{w}_{\boldsymbol{v}_h}, w_{p_h}\right\} \in \mathcal{V}_{\boldsymbol{u}_h} \times \mathcal{V}_{\boldsymbol{v}_h} \times \mathcal{V}_{p_h}$, with $\dot{\boldsymbol{y}}_h(t) := \{d\boldsymbol{u}_h/dt, d\boldsymbol{v}_h/dt, dp_h/dt\}^T$ and $\boldsymbol{y}_h(0) := \{\boldsymbol{u}_{h0}, \boldsymbol{v}_{h0}, p_{h0}\}^T$. Here $\boldsymbol{u}_{h0}$, $\boldsymbol{v}_{h0}$, and $p_{h0}$ are the $\mathcal{L}^2$ projections of the initial data onto the finite dimensional trial solution spaces. In the above and henceforth, the formulations for the kinematic equations, the linear momentum equations, and the mass equation are indicated by the subscripts $k$, $m$, and $p$ respectively.

The terms involving $\boldsymbol{\tau}^e_M$ in (3.3) arise from the subgrid-scale modeling [1]. These terms improve the stability of the Galerkin formulation without sacrificing the consistency. The design of the stabilization parameter $\boldsymbol{\tau}^e_M$ is the crux of the design of the VMS formulation. In this work, the following choices are made,

$$\boldsymbol{\tau}^e_M = \tau^e_M \boldsymbol{I}_{n_{sd}}, \quad \tau^e_M = c_m \frac{h^e}{c\rho}.$$

In the above, $\boldsymbol{I}_{n_{sd}}$ is the second-order identity tensor; $c_m$ is a dimensionless parameter; $c$ is the maximum wave speed in the solid body. For compressible materials, $c$ is given by the bulk wave speed. Under the isotropic small-strain linear elastic assumption, $c = \sqrt{(\lambda+2\mu)/\rho_0}$, where $\lambda$ and $\mu$ are the Lamé parameters. For incompressible materials, $c = \sqrt{\mu/\rho_0}$ is the shear wave speed. We point out that, although the choices made above are based on a simplified material model, the stabilization terms still provide an effective pressure stabilization mechanism for a range of elastic and inelastic problems [1,3,39–41]. In this work, we fix $c_m$ to be $10^{-3}$ and restrict our discussion to the low-order finite element method (i.e. $\mathsf{k}=1$).

### 3.2. Temporal discretization

Based on the semi-discrete formulation (3.1)–(3.3), we invoke the generalized-$\alpha$ method [38] for time integration. The time interval $[0, T]$ is divided into a set of $n_{ts}$ subintervals of size $\Delta t_n := t_{n+1} - t_n$ delimited by a discrete time vector $\{t_n\}_{n=0}^{n_{ts}}$. The solution vector and its first-order time derivative evaluated at the time step $t_n$ are denoted as $\boldsymbol{y}_n$ and $\dot{\boldsymbol{y}}_n$; the basis function for the discrete function spaces is denoted as $N_A$. With these notations, the residual vectors can be represented as

$$\begin{aligned}
\mathbf{R}_k\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) &:= \left\{\mathbf{B}_k\left(N_A \boldsymbol{e}_i; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\}, \\
\mathbf{R}_m\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) &:= \left\{\mathbf{B}_m\left(N_A \boldsymbol{e}_i; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\}, \\
\mathbf{R}_p\left(\dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right) &:= \left\{\mathbf{B}_p\left(N_A; \dot{\boldsymbol{y}}_n, \boldsymbol{y}_n\right)\right\}.
\end{aligned}$$

The fully discrete scheme can be stated as follows. At time step $t_n$, given $\dot{\boldsymbol{y}}_n$, $\boldsymbol{y}_n$, the time step size $\Delta t_n$, and the parameters $\alpha_m$, $\alpha_f$, and $\gamma$, find $\dot{\boldsymbol{y}}_{n+1}$ and $\boldsymbol{y}_{n+1}$ such that

$$\mathbf{R}_k(\dot{\boldsymbol{y}}_{n+\alpha_m}, \boldsymbol{y}_{n+\alpha_f}) = \mathbf{0}, \tag{3.4}$$

$$\mathbf{R}_m(\dot{\boldsymbol{y}}_{n+\alpha_m}, \boldsymbol{y}_{n+\alpha_f}) = \mathbf{0}, \tag{3.5}$$

$$\mathbf{R}_p(\dot{\boldsymbol{y}}_{n+\alpha_m}, \boldsymbol{y}_{n+\alpha_f}) = \mathbf{0}, \tag{3.6}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \dot{\boldsymbol{y}}_n, + \gamma \Delta t_n \left(\dot{\boldsymbol{y}}_{n+1} - \dot{\boldsymbol{y}}_n\right), \tag{3.7}$$

$$\dot{\boldsymbol{y}}_{n+\alpha_m} = \dot{\boldsymbol{y}}_n + \alpha_m \left(\dot{\boldsymbol{y}}_{n+1} - \dot{\boldsymbol{y}}_n\right), \tag{3.8}$$

$$\boldsymbol{y}_{n+\alpha_f} = \boldsymbol{y}_n + \alpha_f \left(\boldsymbol{y}_{n+1} - \boldsymbol{y}_n\right). \tag{3.9}$$

The choice of the parameters $\alpha_m$, $\alpha_f$ and $\gamma$ determines the accuracy and stability of the temporal scheme. Importantly, the high-frequency dissipation can be controlled via a proper parametrization of these parameters, while maintaining second-order accuracy and unconditional stability (for linear problems). For first-order dynamic problems, the parameters are chosen 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 [38]. We adopt $\varrho_\infty = 0.5$ for all computations presented in this work.

**Remark 1.** Interested readers are referred to [42] for the parametrization of the parameters for second-order structural dynamics. A recent study shows that using the generalized-$\alpha$ method for the first-order structural dynamics enjoys improved dissipation and dispersion properties and does not suffer from overshoot [43]. Moreover, using a first-order structural dynamic model is quite propitious for the design of a FSI scheme [1].

### 3.3. A segregated predictor multi-corrector algorithm

One may apply an inverse of the mass matrix at both sides of the equations (3.4) and obtain the following simplified kinematic equations,

$$\overline{\mathbf{R}}_k(\dot{\boldsymbol{y}}_{n+\alpha_m}, \boldsymbol{y}_{n+\alpha_f}) := \dot{\boldsymbol{u}}_{n+\alpha_m} - \boldsymbol{v}_{n+\alpha_f} = \mathbf{0}. \tag{3.10}$$

This procedure can be regarded as the application of a left preconditioner on the nonlinear algebraic equations. The new equations (3.10), together with (3.6) and (3.5), constitute the system of nonlinear algebraic equations to be solved in each time step. The Newton–Raphson method with consistent linearization is invoked to solve the nonlinear system of equations. At the time step $t_{n+1}$, the solution vector $\boldsymbol{y}_{n+1}$ is solved by means of a predictor multi-corrector algorithm. We denote $\boldsymbol{y}_{n+1,(l)} := \left\{\boldsymbol{u}_{n+1,(l)}, \boldsymbol{v}_{n+1,(l)}, p_{n+1,(l)}\right\}^T$ as the solution vector at the Newton–Raphson iteration step $l = 0, \cdots, l_{max}$. The residual vectors evaluated at the iteration stage $l$ are denoted as

$$\begin{aligned}
\mathbf{R}_{(l)} &:= \left\{\overline{\mathbf{R}}_{k,(l)}, \mathbf{R}_{p,(l)}, \mathbf{R}_{m,(l)}\right\}^T, \\
\overline{\mathbf{R}}_{k,(l)} &:= \overline{\mathbf{R}}_k\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right), \\
\mathbf{R}_{m,(l)} &:= \mathbf{R}_m\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right), \\
\mathbf{R}_{p,(l)} &:= \mathbf{R}_p\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right).
\end{aligned}$$

The consistent tangent matrix associated with the above residual vectors is

$$\mathbf{K}_{(l)} = \begin{bmatrix} \mathbf{K}_{k,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{k,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{k,(l),\dot{p}} \\ \mathbf{K}_{m,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{m,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{m,(l),\dot{p}} \\ \mathbf{K}_{p,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{p,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{p,(l),\dot{p}} \end{bmatrix},$$

wherein

$$\mathbf{K}_{k,(l),\dot{\boldsymbol{u}}} := \alpha_m \frac{\partial \overline{\mathbf{R}}_{k,(l)}\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \dot{\boldsymbol{u}}_{n+\alpha_m}} = \alpha_m \mathbf{I},$$

$$\mathbf{K}_{k,(l),\dot{\boldsymbol{v}}} := \alpha_f \gamma \Delta t_n \frac{\partial \overline{\mathbf{R}}_{k,(l)}\left(\dot{\boldsymbol{y}}_{n+\alpha_m,(l)}, \boldsymbol{y}_{n+\alpha_f,(l)}\right)}{\partial \boldsymbol{v}_{n+\alpha_f}} = -\alpha_f \gamma \Delta t_n \mathbf{I},$$

$$\mathbf{K}_{k,(l),\dot{p}} := \mathbf{O}.$$

As was realized in [3], this special block structure in the first row of $\mathbf{K}_{(l)}$ can be utilized for a block factorization,

$$\begin{aligned} \mathbf{K}_{(l)} &= \begin{bmatrix} \mathbf{K}_{k,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{k,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{k,(l),\dot{p}} \\ \mathbf{K}_{m,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{m,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{m,(l),\dot{p}} \\ \mathbf{K}_{p,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{p,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{p,(l),\dot{p}} \end{bmatrix} = \begin{bmatrix} \alpha_m \mathbf{I} & -\alpha_f \gamma \Delta t_n \mathbf{I} & \mathbf{O} \\ \mathbf{K}_{m,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{m,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{m,(l),\dot{p}} \\ \mathbf{K}_{p,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{p,(l),\dot{\boldsymbol{v}}} & \mathbf{K}_{p,(l),\dot{p}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{O} & \mathbf{O} \\ \frac{1}{\alpha_m}\mathbf{K}_{m,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{m,(l),\dot{\boldsymbol{v}}} + \frac{\alpha_f \gamma \Delta t_n}{\alpha_m}\mathbf{K}_{m,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{m,(l),\dot{p}} \\ \frac{1}{\alpha_m}\mathbf{K}_{p,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{p,(l),\dot{\boldsymbol{v}}} + \frac{\alpha_f \gamma \Delta t_n}{\alpha_m}\mathbf{K}_{p,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{p,(l),\dot{p}} \end{bmatrix} \begin{bmatrix} \alpha_m \mathbf{I} & -\alpha_f \gamma \Delta t_n \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{I} \end{bmatrix}. \end{aligned} \tag{3.11}$$

With (3.11), the solution procedure of the linear system of equations in the Newton–Raphson method can be consistently reduced to a two-stage algorithm [1,3,44]. In the first stage, one obtains the increments of the pressure and velocity at the iteration step $l$ by solving the following linear system,

$$\begin{bmatrix} \mathbf{K}_{m,(l),\dot{\boldsymbol{v}}} + \frac{\alpha_f \gamma \Delta t_n}{\alpha_m}\mathbf{K}_{m,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{m,(l),\dot{p}} \\ \mathbf{K}_{p,(l),\dot{\boldsymbol{v}}} + \frac{\alpha_f \gamma \Delta t_n}{\alpha_m}\mathbf{K}_{p,(l),\dot{\boldsymbol{u}}} & \mathbf{K}_{p,(l),\dot{p}} \end{bmatrix} \begin{bmatrix} \Delta \dot{\boldsymbol{v}}_{n+1,(l)} \\ \Delta \dot{p}_{n+1,(l)} \end{bmatrix} = - \begin{bmatrix} \mathbf{R}_{m,(l)} - \frac{1}{\alpha_m}\mathbf{K}_{m,(l),\dot{\boldsymbol{u}}}\overline{\mathbf{R}}_{k,(l)} \\ \mathbf{R}_{p,(l)} - \frac{1}{\alpha_m}\mathbf{K}_{p,(l),\dot{\boldsymbol{u}}}\overline{\mathbf{R}}_{k,(l)} \end{bmatrix}. \tag{3.12}$$

In the second stage, one obtains the increments for the displacement by

$$\Delta \dot{\boldsymbol{u}}_{n+1,(l)} = \frac{\alpha_f \gamma \Delta t_n}{\alpha_m} \Delta \dot{\boldsymbol{v}}_{n+1,(l)} - \frac{1}{\alpha_m} \overline{\mathbf{R}}^k_{(l)}. \tag{3.13}$$

To simplify notations in the following discussion, we denote

$$\mathbf{A}_{(l)} := \mathbf{K}_{m,(l),\dot{\boldsymbol{v}}} + \frac{\alpha_f \gamma \Delta t_n}{\alpha_m}\mathbf{K}_{m,(l),\dot{\boldsymbol{u}}}, \quad \mathbf{B}_{(l)} := \mathbf{K}_{m,(l),\dot{p}}, \tag{3.14}$$

$$\mathbf{C}_{(l)} := \mathbf{K}_{p,(l),\dot{\boldsymbol{v}}} + \frac{\alpha_f \gamma \Delta t_n}{\alpha_m}\mathbf{K}_{p,(l),\dot{\boldsymbol{u}}}, \quad \mathbf{D}_{(l)} := \mathbf{K}_{p,(l),\dot{p}}. \tag{3.15}$$

**Remark 2.** In [1], it was shown that $\overline{\mathbf{R}}^k_{(l)} = \mathbf{0}$ for $l \geq 2$ for general predictor multi-corrector algorithms; in [44], a special predictor is chosen so that $\overline{\mathbf{R}}^k_{(l)} = \mathbf{0}$ for $l \geq 1$.

**Remark 3.** In Appendix A, the detailed formula for the block matrices are given, and it can be observed that $\mathbf{A}_{(l)}$ consists primarily of a mass matrix and a stiffness matrix; $\mathbf{B}_{(l)}$ is a discrete gradient operator; $\mathbf{C}_{(l)}$ is dominated by a discrete divergence operator; $\mathbf{D}_{(l)}$ contains a mass matrix scaled with $\beta$ and contributions from the stabilization terms.

Based on the above discussion, a 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. Evaluate the solution vectors at the intermediate stages:

$$\boldsymbol{y}_{n+\alpha_f,(l)} = \boldsymbol{y}_n + \alpha_f \left(\boldsymbol{y}_{n+1,(l-1)} - \boldsymbol{y}_n\right), \quad \dot{\boldsymbol{y}}_{n+\alpha_m,(l)} = \dot{\boldsymbol{y}}_n + \alpha_m \left(\dot{\boldsymbol{y}}_{n+1,(l-1)} - \dot{\boldsymbol{y}}_n\right).$$

2. Assemble the residual vectors $\mathbf{R}_{m,(l)}$ and $\mathbf{R}_{p,(l)}$ using the solution evaluated at the intermediate stages.
3. Let $\|\mathbf{R}_{(l)}\|_{l^2}$ denote the $l^2$-norm of the residual vector. If either one of the following stopping criteria

$$\frac{\|\mathbf{R}_{(l)}\|_{l^2}}{\|\mathbf{R}_{(0)}\|_{l^2}} \leq \text{tol}_\text{R}, \qquad \|\mathbf{R}_{(l)}\|_{l^2} \leq \text{tol}_\text{A},$$

is satisfied for two prescribed tolerances $\text{tol}_\text{R}$, $\text{tol}_\text{A}$, 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 4.
4. Assemble the tangent matrices (3.14)–(3.15).
5. Solve the following linear system of equations for $\Delta\dot{p}_{n+1,(l)}$ and $\Delta\dot{\boldsymbol{v}}_{n+1,(l)}$,

$$\begin{bmatrix} \mathbf{A}_{(l)} & \mathbf{B}_{(l)} \\ \mathbf{C}_{(l)} & \mathbf{D}_{(l)} \end{bmatrix} \begin{bmatrix} \Delta\dot{\boldsymbol{v}}_{n+1,(l)} \\ \Delta\dot{p}_{n+1,(l)} \end{bmatrix} = - \begin{bmatrix} \mathbf{R}_{m,(l)} \\ \mathbf{R}_{p,(l)} \end{bmatrix}. \tag{3.16}$$

6. Obtain $\Delta\dot{\boldsymbol{u}}_{n+1,(l)}$ from the relation (3.13).
7. Update the solution vector 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 all the numerical simulations presented in this work, 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 $l_{max} = 20$.

## 4. Iterative linear solver

In the predictor multi-corrector algorithm presented above, the linear system of equations (3.16) is solved repeatedly, and this step constitutes the major cost for implicit dynamic calculations. In this section, we design an iterative solution procedure for the linear problem $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$, in which the matrix and vectors adopt the following block structure,

$$\mathcal{A} := \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \quad \boldsymbol{x} := \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{v}} \\ \boldsymbol{x}_p \end{bmatrix}, \quad \boldsymbol{r} := \begin{bmatrix} \boldsymbol{r}_{\boldsymbol{v}} \\ \boldsymbol{r}_p \end{bmatrix}.$$

Since its inception, GMRES is among the most popular iterative methods for solving sparse nonsymmetric matrix problems. With a proper preconditioner $\mathcal{P}$, the convergence rate of iterative methods like GMRES can be significantly expedited. Roughly speaking, in the GMRES iteration, one constructs the Krylov subspace and search for the solution that minimize the residual in this Krylov subspace by the Arnoldi algorithm [33,34]. To construct the Krylov subspace, one applies $\mathcal{A}\mathcal{P}^{-1}$ to the residual vector in order to enlarge the Krylov subspace. This procedure corresponds to first solving a linear system of equations associated with $\mathcal{P}$ and then performing a matrix–vector multiplication associated with $\mathcal{A}$. Often times, to reduce the computational burden, the GMRES algorithm is restarted every $\mathfrak{m}$ steps. Within this work, this algorithm is denoted as GMRES($\mathfrak{m}$).

In Section 4.1, we perform a diagonal scaling for $\mathcal{A}$ with the purpose of improving the condition number [24,31,45]. In Section 4.2, we introduce the block factorization of $\mathcal{A}$ and present the SCR algorithm. In Section 4.3, we present the coupled approach with a particular focus on the SIMPLE preconditioner. In Section 4.4, the nested block preconditioning technique is introduced as a combination of the SCR approach and the coupled approach.

### 4.1. Symmetrically diagonal scaling

Before constructing an iterative method, we first apply a symmetrically diagonal scaling to the matrix $\mathcal{A}$. This approach is adopted to improve the condition number of the matrix problem and is sometimes referred to as a "pre-preconditioning" technique [45]. We introduce $\mathcal{W}$ as a diagonal matrix defined as follows,

$$\mathcal{W}_{ii} := \begin{cases} (|\mathcal{A}_{ii}|)^{-\frac{1}{2}}, & \text{if } |\mathcal{A}_{ii}| \geq \epsilon_{diag} \\ 1.0, & \text{if } |\mathcal{A}_{ii}| < \epsilon_{diag} \end{cases}.$$

In the above definition, $\epsilon_{diag}$ is a user-specified tolerance to avoid undefined or unstable numerical operations. In this work, we set $\epsilon_{diag} = 1.0 \times 10^{-15}$. Applying $\mathcal{W}$ as a left and right preconditioner simultaneously, we obtain an altered system as

$$\mathcal{A}^*\boldsymbol{x}^* = \boldsymbol{r}^*, \tag{4.1}$$

wherein $\mathcal{A}^* := \mathcal{W}\mathcal{A}\mathcal{W}$, $\boldsymbol{x}^* := \mathcal{W}^{-1}\boldsymbol{x}$, and $\boldsymbol{r}^* = \mathcal{W}\boldsymbol{r}$. The iterative methods discussed in the subsequent sections are applied to the above system. Once $x^*$ is obtained from (4.1), one has to perform $\boldsymbol{x} = \mathcal{W}\boldsymbol{x}^*$ to recover the true solution. In the remainder of Section 4, we focus on solving (4.1), and for notational simplicity, the superscript $*$ is neglected.

### 4.2. Schur complement reduction

Recall that $\mathcal{A}$ adopts the block factorization

$$\mathcal{A} = \mathcal{L}\mathcal{D}\mathcal{U} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{O} \\ \mathbf{O} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}, \tag{4.2}$$

wherein $\mathbf{S} := \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ is the Schur complement of $\mathbf{A}$. Applying $\mathcal{L}^{-1}$ on both sides of the equation $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$, one obtains

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{O} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{v}} \\ \boldsymbol{x}_p \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{r}_{\boldsymbol{v}} \\ \boldsymbol{r}_p \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{\boldsymbol{v}} \\ \boldsymbol{r}_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{\boldsymbol{v}} \\ \boldsymbol{r}_p - \mathbf{C}\mathbf{A}^{-1}\boldsymbol{r}_{\boldsymbol{v}} \end{bmatrix}.$$

The upper triangular block matrix problem can be solved by a back substitution. Consequently, the solution procedure for $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$ can be summarized as the following segregated algorithm [24–26].

**Algorithm 1** Solution procedure for $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$ based on SCR.

1: Solve for an intermediate velocity $\hat{\boldsymbol{x}}_{\boldsymbol{v}}$ from the equation

$$\mathbf{A}\hat{\boldsymbol{x}}_{\boldsymbol{v}} = \boldsymbol{r}_{\boldsymbol{v}}. \tag{4.3}$$

2: Update the continuity residual by $\boldsymbol{r}_p \leftarrow \boldsymbol{r}_p - \mathbf{C}\hat{\boldsymbol{x}}_{\boldsymbol{v}}$.
3: Solve for $\boldsymbol{x}_p$ from the equation

$$\mathbf{S}\boldsymbol{x}_p = \boldsymbol{r}_p. \tag{4.4}$$

4: Update the momentum residual by $\boldsymbol{r}_{\boldsymbol{v}} \leftarrow \boldsymbol{r}_{\boldsymbol{v}} - \mathbf{B}\boldsymbol{x}_p$.
5: Solve for $\boldsymbol{x}_{\boldsymbol{v}}$ from the equation

$$\mathbf{A}\boldsymbol{x}_{\boldsymbol{v}} = \boldsymbol{r}_{\boldsymbol{v}}. \tag{4.5}$$

For hyper-elastodynamics problems, it is reasonable to apply GMRES preconditioned by AMG for (4.3) and (4.5). The stopping condition for solving with $\mathbf{A}$ includes the tolerance for the relative error $\delta_A^r$, the tolerance for the absolute error $\delta_A^a$, and the maximum number of iterations $n_A^{max}$. In (4.4), the Schur complement is a dense matrix due to the presence of $\mathbf{A}^{-1}$ in its definition. It is expensive and often impossible to directly compute with $\mathbf{S}$. Recall that in a Krylov subspace method, the search space is iteratively expanded by performing matrix–vector multiplications. Although the algebraic form of $\mathbf{S}$ is impractical to obtain, its action on a vector is readily available through the following "matrix-free" algorithm [24,26].

**Algorithm 2** 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

$$\mathbf{A}\tilde{\boldsymbol{x}}_p = \bar{\boldsymbol{x}}_p. \tag{4.6}$$

4: Compute the matrix–vector multiplication $\bar{\boldsymbol{x}}_p \leftarrow \mathbf{C}\tilde{\boldsymbol{x}}_p$.
5: **return** $\hat{\boldsymbol{x}}_p - \bar{\boldsymbol{x}}_p$.

In Algorithm 2, the action of $\mathbf{S}$ on a vector is realized through a series of matrix–vector multiplications, and the action of $\mathbf{A}^{-1}$ on a vector is achieved by solving the linear system (4.6). This solver is located inside the solution procedure of (4.4), and we call it the *inner solver*. The stopping condition of the inner solver includes the tolerance for the relative error $\delta_I^r$, the tolerance for the absolute error $\delta_I^a$, and the maximum number of iterations $n_I^{max}$.

With Algorithm 2, one can construct a Krylov subspace for $\mathbf{S}$ and solve the equation (4.4). However, without preconditioning, GMRES may stagnate or even break down. More importantly, each matrix–vector multiplication given in Algorithm 2 involves solving a linear system (4.6), and this inevitably makes the matrix–vector multiplication quite expensive. To mitigate the number of this expensive matrix–vector multiplications, we solve (4.4) with $\hat{\mathbf{S}} := \mathbf{D} - \mathbf{C}\left(\mathrm{diag}\left(\mathbf{A}\right)\right)^{-1}\mathbf{B}$ as a right preconditioner [46]. If the time step size is small, $\mathbf{A}$ is dominated by the mass matrix, and $\hat{\mathbf{S}}$ acts as an effective preconditioner for solving (4.4). On the other side, if the time step size is large, $\mathbf{A}$ is dominated by the stiffness matrix. The situation

then is analogous to the Stokes problem, where the Schur complement is spectrally equivalent to an identity matrix. We may reasonably expect that an unpreconditioned GMRES using Algorithm 2 is sufficient for solving (4.4). Still, using $\hat{\mathbf{S}}$ may accelerate the convergence rate. Therefore, we solve (4.4) by GMRES, where the stopping criteria include the tolerance for the relative error $\delta_S^r$, the tolerance for the absolute error $\delta_S^a$, and the maximum number of iterations $n_S^{max}$.

### 4.3. Coupled approach with block preconditioners

The block factorization (4.2) also inspires the design of a preconditioner for $\mathcal{A}$. Following the nomenclature used in [16], we use $\mathbf{H}_1$ and $\mathbf{H}_2$ to denote the approximations of $\mathbf{A}^{-1}$ in the Schur complement and the upper triangular matrix $\mathcal{U}$, respectively. This results in a block preconditioner expressed as

$$\hat{\mathcal{P}} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{O} \\ \mathbf{O} & \mathbf{D} - \mathbf{C}\mathbf{H}_1\mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{H}_2\mathbf{B} \\ \mathbf{O} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{A}\mathbf{H}_2\mathbf{B} \\ \mathbf{C} & \mathbf{D} - \mathbf{C}\left(\mathbf{H}_1 - \mathbf{H}_2\right)\mathbf{B} \end{bmatrix}. \tag{4.7}$$

The two approximated sparse matrices are introduced so that the spectrum of $\mathcal{A}\hat{\mathcal{P}}^{-1}$ has a clustering around $\{1\}$. With the block preconditioner, one can apply the Krylov subspace method directly to solve $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$, and the bases of the Krylov subspace are constructed by applying $\mathcal{A}\hat{\mathcal{P}}^{-1}$ on a vector. The action of $\hat{\mathcal{P}}^{-1}$ is achieved through a procedure similar to the Algorithm 1. The differences are that the inner solver is not needed and one does not need to solve the equations associated with the sub-matrices to a high precision. The Krylov subspace method is typically used with a multigrid [17,32] or a domain decomposition [47] preconditioner to solve with the sub-matrices. Consequently, the algebraic definition of $\hat{\mathcal{P}}$ varies over iterations, and one has to apply a flexible method, like the Flexible GMRES (FGMRES) [34], as the iterative method for $\mathcal{A}$. Choosing $\mathbf{H}_1 = \mathbf{H}_2 = \mathrm{diag}\left(\mathbf{A}\right)^{-1}$ leads to the SIMPLE preconditioner $\hat{\mathcal{P}}_{\mathrm{SIMPLE}}$ [17,16],

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

The SIMPLE preconditioner is an algebraic analogue of the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) [18]. It introduces a perturbation to the pressure operator in the linear momentum equation. This preconditioner and its variants are among the most popular choices for problems in CFD [32,48], FSI [47], and multiphysics problems [49,50].

**Remark 4.** There are cases when the symmetry of $\mathbf{A}$ is broken, and using the SIMPLE-type preconditioner leads to poor performance. It is the case in CFD with large Reynolds numbers. To take into account of the off-diagonal entries of $\mathbf{A}$, sophisticated preconditioners, like the LSC preconditioner [22], have been developed. Those preconditioners have been shown to be robust with respect to the Reynolds number using inf-sup stable discretizations of the CFD problem (i.e., $\mathbf{D} = \mathbf{O}$). Note that, for the stabilized methods, the LSC preconditioner may not converge [32].

### 4.4. Flexible GMRES algorithm with a nested block preconditioner

The SIMPLE preconditioner can be viewed as the SCR approach built based on an inexact block factorization. Its main advantage is that the application of this preconditioner is inexpensive. However, for certain problems, this inexact factorization misses some key information of the original matrix, and stagnation of the solver is observed. We want to leverage the robustness of the SCR approach built from the exact block factorization by using it as a right preconditioner, denoted as $\hat{\mathcal{P}}_{SCR}$. The action of $\hat{\mathcal{P}}_{SCR}^{-1}$ on a vector is given by Algorithm 1, in which the equations (4.3)–(4.5) are solved with prescribed tolerances. The algebraic form of $\hat{\mathcal{P}}_{SCR}$ is defined implicitly through the solvers in Algorithm 1 and varies over iterations. Assuming that the three equations (4.3)–(4.5) are solved exactly, the spectrum of $\mathcal{A}\hat{\mathcal{P}}_{SCR}^{-1}$ will be $\{1\}$, and the solver will converge in one iteration. Because the preconditioner varies over iterations, we invoke FGMRES as the iterative method for $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$. The stopping condition of the FGMRES algorithm includes the tolerance for the absolute error $\delta^a$, the tolerance for the relative error $\delta^r$, and the maximum number of iterations $n^{max}$.

The FGMRES iteration for $\mathcal{A}$ serves as the *outer solver* which tries to minimize the residual of $\mathcal{A}\boldsymbol{x} = \boldsymbol{r}$. Inside this FGMRES iteration, the application of $\hat{\mathcal{P}}_{SCR}$ is achieved through Algorithm 1, and one needs to solve with the block matrices $\mathbf{A}$ and $\mathbf{S}$ at this stage. We call it the *intermediate solver*. When solving with the Schur complement, its action on a vector is defined by Algorithm 2, which necessitates using the *inner solver* to solve with $\mathbf{A}$. The three levels of solvers are illustrated in Fig. 1 with different colors.

**Remark 5.** In the construction of the proposed block preconditioners, the full $\mathcal{LDU}$ factorization of $\mathcal{A}$ is utilized. One can surely use only part of the factorization to devise different preconditioners. For example, the diagonal part $\mathcal{D}$ is an efficient candidate for the Stokes equations [19,51]. Assuming exact arithmetic, it gives convergence within 4 iterations. Using the upper triangular part $\mathcal{DU}$ often gives a good balance between the convergence rate and the computational cost [52], as it leads to convergence within 2 iterations [53,54], assuming exact arithmetic. In our case, the full $\mathcal{LDU}$ block factorization gives the fastest convergence rate. We prefer this because the solution of the Schur complement equation is often the most

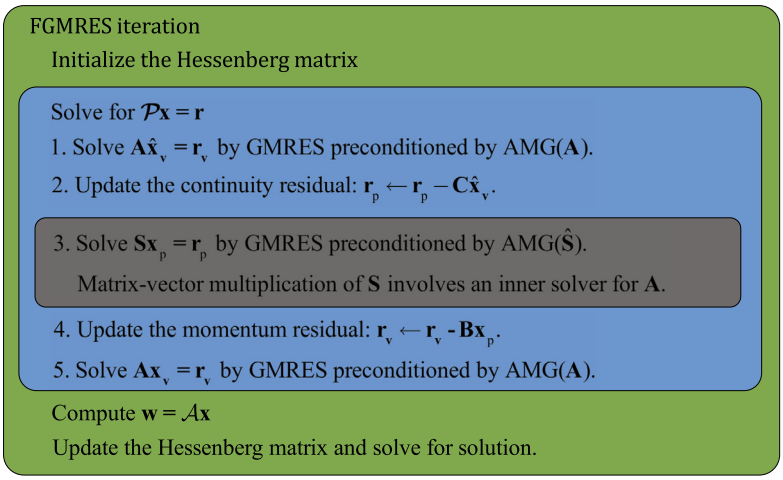


**Fig. 1.** Implementation of the FGMRES with the nested block preconditioner. The green color represents the outer solver; the blue color represents the intermediate solver; the grey color represents the inner solver. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

expensive part of the overall algorithm. Therefore, in comparison with an upper triangular block preconditioner, we pay the price of solving the matrix problem $\mathbf{A}$ twice with the purpose of mitigating the number of the solution procedure for the Schur complement.

**Remark 6.** In the above algorithm, the nested block preconditioner $\hat{\mathcal{P}}_{SCR}$ can be regarded as a result of an inexact factorization of $\mathcal{A}$. The inexactness is due to the approximation made by the solvers in the intermediate and inner levels. The preconditioner is thus defined by the tolerances of these solvers. Using strict tolerances apparently makes $\hat{\mathcal{P}}_{SCR}$ closer to $\mathcal{A}$. However, this is impractical since this makes the algorithm as expensive as the SCR approach. On the other extreme, one may solve (4.4) by applying the preconditioner $\hat{\mathbf{S}}$ once without invoking the inner solver. This makes the algorithm as simple as the coupled approach with the SIMPLE preconditioner and potentially endangers the robustness. We adjust the tolerances to tune the preconditioner, noting there is a lot of leeway in the choice of the tolerance value ranging from strict to loose. The effect of the tolerances of the intermediate and inner solvers will be studied in Section 5.

**Remark 7.** Choosing a good preconditioner for the Schur complement is critical for the performance of the proposed nested block preconditioner. In our experience, using a scaled pressure mass matrix gives satisfactory results as well [55]. For compressible materials, this preconditioner does not need to be explicitly assembled, and one can use $\mathbf{D}$ directly (See Appendix A). In this work, we focus on $\mathbf{D} - \mathbf{C}\left(\text{diag}\left(\mathbf{A}\right)\right)^{-1}\mathbf{B}$, since this choice apparently is a better approximation of $\mathbf{S}$. In [56], a sparse approximate inverse is utilized to construct the preconditioner for the Schur complement, which is worth of future study.

## 5. Numerical results

In our work, the outer solver is FGMRES(200) with $n^{max} = 200$ and $\delta^a = 10^{-50}$. In the intermediate level, (4.3) and (4.5) are solved by GMRES(500) preconditioned by AMG with $n_A^{max} = 500$ and $\delta_A^a = 10^{-50}$. The equation (4.4) is solved by GMRES(200), with $n_S^{max} = 200$ and $\delta_S^a = 10^{-50}$. We use the AMG preconditioner constructed from $\hat{\mathbf{S}}$. In the inner level, the linear system is solved via GMRES(500) preconditioned by AMG with $n_I^{max} = 500$ and $\delta_I^a = 10^{-50}$. We use the BoomerAMG [57] from the Hypre package [58] as the parallel AMG implementation. The settings of the BoomerAMG are summarized in Table 1. With the above settings, the accuracy of the solution is dictated by $\delta^r$, and the convergence rate is controlled by the tolerances $\delta_A^r$, $\delta_S^r$, and $\delta_I^r$.

To provide baseline examples, we solve the system of equations (4.1) by two different preconditioners. As the first example, we solve the system of equations by FGMRES(200) using $\hat{\mathcal{P}}_{\text{SIMPLE}}$ with $n^{max} = 200$ and $\delta^a = 10^{-50}$. In this preconditioner, the settings of the linear solver (including the Krylov subspace method, the preconditioners, and the stopping criteria) associated with $\mathbf{A}$ and $\hat{\mathbf{S}}$ are exactly the same as the ones used in the nested block preconditioner. The accuracy of the solver is determined by $\delta^r$, and the performance of the preconditioner is controlled by $\delta_A^r$ and $\delta_S^r$. Notice that, in this preconditioner, $\delta_S^r$ is the tolerance for solving with the matrix $\hat{\mathbf{S}}$.

As another baseline example, we choose to solve the linear system by GMRES(200) preconditioned by a one-level additive Schwarz domain decomposition preconditioner [59]. The maximum number of iterations is fixed at 10000, and the tolerance for the absolute error is fixed at $10^{-50}$. In this preconditioner, each processor is assigned with a single subdomain,

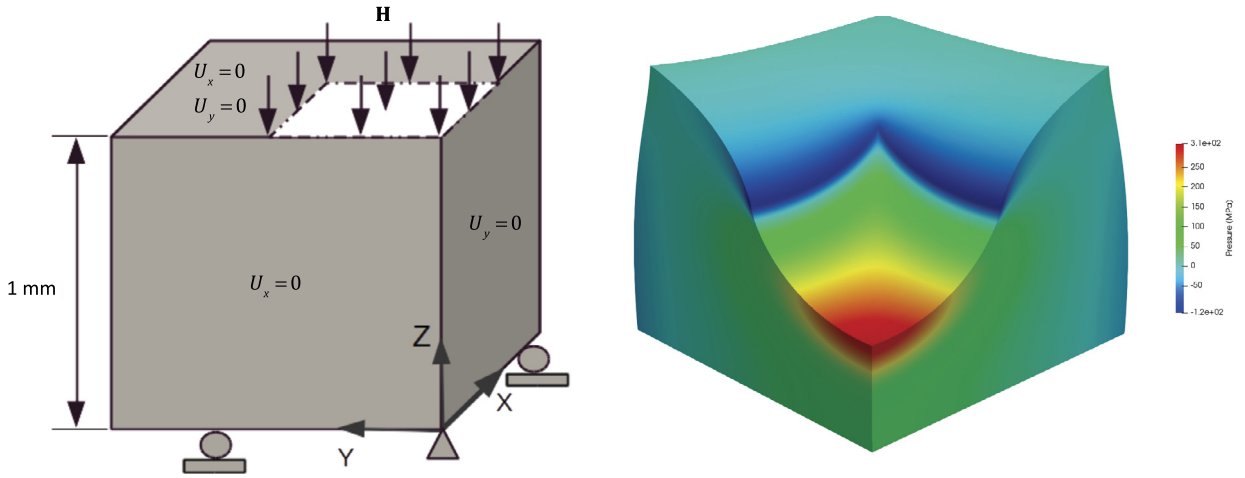


**Fig. 2.** Three-dimensional compression of a block: (left) geometry of the referential configuration and the boundary conditions; (right) pressure profile in the current configuration with $\Delta x = 1/3840$.

**Table 1**
Settings of the BoomerAMG preconditioner [58].

| | |
|---|---|
| Cycle type | V-cycle |
| Coarsening method | HMIS |
| Interpolation method | Extended method (ext+i) |
| 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 |

and an incomplete LU factorization (ILU) with a fill-in ratio 1.0 is invoked to solve the problem on the subdomains. This preconditioner is purely algebraic and is usually very competitive for medium-size parallel simulations. However, as will be shown in the numerical examples, the one-level domain decomposition preconditioner is not a robust option. Also, as the problem size and the number of subdomains grows, more iterations are needed to propagate information across the whole domain. In our implementation, the restricted additive Schwarz method from PETSc [60] is utilized as the domain decomposition preconditioner; the PILUT routine from Hypre [58] is used as the solver for the subdomain algebraic problem.

All numerical simulations are performed on the Stampede2 supercomputer at Texas Advanced Computing Center (TACC), using the Intel Xeon Platinum 8160 node. Each node contains 48 cores, with 2.1 GHz nominal clock rate and 192 GB RAM per node (4 GB RAM per core).

### 5.1. Compression of a block

The compression of a unit block was proposed as a benchmark problem for nearly incompressible solids [36]. The geometrical configuration and the boundary conditions are illustrated in Fig. 2. The problem is discretized in space by a uniform structured tetrahedral mesh generated by Gmsh [61], and we use $\Delta x$ to denote the edge length of the mesh. The original benchmark problem was proposed in the quasi-static setting, and a 'dead' surface load $\boldsymbol{H}$ is applied on a quarter portion of the top surface, pointing in the negative $z$-direction with magnitude $|\boldsymbol{H}| = 320$ MPa. In this work, the problem is investigated in the dynamic setting by gradually increasing the load force as a linear function of time. The material is described by a Neo-Hookean model, whose Gibbs free energy function takes the form

$$G\left(\tilde{\boldsymbol{C}}, p\right) = \frac{\mu}{2\rho_0}\left(\mathrm{tr}\tilde{\boldsymbol{C}} - 3\right) + \frac{p\sqrt{p^2+\kappa^2} - p^2}{2\kappa\rho_0} - \frac{\kappa}{2\rho_0}\ln\left(\frac{\sqrt{p^2+\kappa^2} - p}{\kappa}\right).$$

Following [36], the material parameters are chosen as $\mu = 80.194$ MPa, $\kappa = 400889.806$ MPa, and $\rho_0 = 1.0 \times 10^3$ kg/m$^3$. The corresponding Poisson's ratio $\nu$ is 0.4999. In Section 5.1.3, we examine the robustness of the preconditioner with regard to varying material moduli. In the following discussion, the governing equations have been non-dimensionalized by the centimetre-gram-second units. Note that the edge length of the cube is 1 mm = 0.1 cm. Then the number of elements in each direction of the cube is given by $1/(10\Delta x)$.

**Table 2**
The impact of the accuracy of the inner solver on the performance of the linear solver. The CPU time is collected for the linear solver only; $\hat{l}$ represents the total number of nonlinear iterations; $n$ represents the total number of FGMRES iterations; $\bar{n}_A$ represents the averaged number of iterations for solving with $\mathbf{A}$ in (4.3) and (4.5); $\bar{n}_S$ represents the averaged number of iterations for solving (4.4); $\bar{n}_I$ represents the averaged number of iterations for solving (4.6).

| | $\delta_I^r$ | CPU time (sec.) | $\hat{l}$ | $n$ | $\bar{n}_A$ | $\bar{n}_S$ | $\bar{n}_I$ |
|---|---|---|---|---|---|---|---|
| $\delta_A^r = \delta_S^r = 10^{-10}$ | $10^0$ | $4.86 \times 10^3$ | 4 | 477 | 74.52 | 33.89 | – |
| | $10^{-2}$ | $9.02 \times 10^2$ | 4 | 17 | 75.62 | 22.29 | 29.31 |
| | $10^{-4}$ | $8.08 \times 10^2$ | 4 | 11 | 75.30 | 22.27 | 45.00 |
| | $10^{-6}$ | $6.97 \times 10^2$ | 4 | 8 | 75.19 | 23.13 | 55.82 |
| | $10^{-8}$ | $8.11 \times 10^2$ | 4 | 8 | 75.19 | 22.13 | 65.15 |
| | $10^{-10}$ | $8.47 \times 10^2$ | 4 | 7 | 74.86 | 23.29 | 74.62 |
| $\delta_A^r = \delta_S^r = 10^{-6}$ | $10^0$ | $4.87 \times 10^3$ | 4 | 664 | 55.60 | 21.29 | – |
| | $10^{-2}$ | $6.30 \times 10^2$ | 4 | 18 | 56.68 | 13.74 | 30.35 |
| | $10^{-4}$ | $5.10 \times 10^2$ | 4 | 11 | 56.30 | 13.73 | 46.14 |
| | $10^{-6}$ | $5.12 \times 10^2$ | 4 | 9 | 56.06 | 14.11 | 56.91 |
| | $10^{-8}$ | $6.01 \times 10^2$ | 4 | 9 | 56.17 | 14.44 | 65.62 |
| | $10^{-10}$ | $6.82 \times 10^2$ | 4 | 9 | 56.17 | 14.56 | 74.87 |

*5.1.1. Performance with varying inner solver accuracy*

In this test, we investigate the impact of the accuracy of the inner solver on the overall iterative method. We fix the mesh size to be $\Delta x = 1/640$ and the time step size to be $\Delta t = 10^{-1}$. The simulation is performed with 8 CPUs, with approximately 131072 equations assigned to each CPU. In this study, we choose $\delta^r = 10^{-8}$, and we consider two settings for the intermediate solver: $\delta_A^r = \delta_S^r = 10^{-10}$ and $\delta_A^r = \delta_S^r = 10^{-6}$. We collect the statistics of the solver in the first time step with varying values of $\delta_I^r$ (Table 2). The results associated with $\delta_I^r = 10^0$ are obtained by solving $\hat{\mathbf{S}}\boldsymbol{x}_p = \hat{\boldsymbol{r}}_p$ in step 3 of Algorithm 1. This choice corresponds to choosing $\mathbf{H}_1 = \mathrm{diag}(\mathbf{A})^{-1}$ and $\mathbf{H}_2 = \mathbf{A}^{-1}$ in (4.7) for $\hat{\mathcal{P}}$, making it similar to the SIMPLE preconditioner.

In our numerical experiments, we observe that with the choice of $\delta_A^r = \delta_S^r = \delta_I^r = 10^{-10}$, the outer solver converges in less than two iterations on average. In fact, we also experimented with stricter tolerances and observed convergence of the outer solver in one iteration. (We do not report this because this stricter choice requires larger size of the Krylov subspace which is incompatible with our current settings.) This result corroborates the fact that the full $\mathcal{LDU}$ block preconditioner gives convergence in one iteration with exact arithmetic.

In the literature, the choice for the inner solver accuracy is under debate. In [24], it is suggested that the inner solver should be more accurate than its upper-level counterpart (i.e., $\delta_I^r \leq \delta_S^r$ in our case) to guarantee accurate representation of the Schur complement. Meanwhile, it is shown in [62] that the Krylov methods are in fact very robust under the presence of inexact matrix–vector multiplications. In our test, as we gradually release the tolerance $\delta_I^r$, it is observed that the inner solver converges with fewer iterations while the outer solver requires more iterations to reach convergence to compensate for the inaccurate evaluations of the Schur complement. As $\delta_I^r$ gets larger than $\delta_S^r$, initially the overhead is low. As the tolerance further increases, the outer solver requires more iterations and the overall cost of the solver grows correspondingly. For the two cases, the break-even points are achieved with $\delta_I^r = 10^{-6}$ and $10^{-4}$, respectively. Examining the number of iterations for the outer solver, we observe a steady growth of $n$ once $\delta_I^r$ grows larger than $\delta_S^r$. Though it is hard to predict the optimal choice of $\delta_I^r$ for general cases, we observe that a choice of $\delta_I^r = \delta_S^r$ is safe for robust performances; a slightly relaxed tolerance for the inner solver (e.g. $\delta_I^r = 10^2 \delta_S^r$) is beneficial for efficiency. We also note that $\bar{n}_S$ is insensitive to the inner solver accuracy as long as $\delta_I^r < 10^0$.

For comparison, we also examined the solver performance without the inner solver. We solve with $\hat{\mathbf{S}}$ instead of $\mathbf{S}$ in (4.4) directly. This corresponds to a highly inaccurate evaluation of the Schur complement. We see that the iteration number $n$, the averaged iteration number for the intermediate solver $\bar{n}_S$, and the CPU time of the outer solver increase significantly. The severe degradation of solver performance signifies the importance of an accurate evaluation of the Schur complement.

*5.1.2. Performance with varying intermediate solver accuracy*

In this example, we examine the effect of varying intermediate solver tolerances on the solver performance. We consider a uniform mesh with $\Delta x = 1/640$, with two time step sizes: $\Delta t = 10^{-1}$ and $10^{-5}$. We choose $\delta^r = 10^{-8}$ for the outer solver. We set $\delta_A^r = \delta_S^r = \delta_I^r$ and vary their values from $10^{-8}$ to $10^{-2}$. To make comparisons, the same problem is simulated with the SIMPLE preconditioner and the additive Schwarz preconditioner. In the SIMPLE preconditioner, we solve the equations associated with $\mathbf{A}$ and $\hat{\mathbf{S}}$ with $\delta_A^r = \delta_S^r = 10^{-8}$. The convergence is monitored for the first time step, which is usually the most challenging part of dynamic calculations. The convergence history of the linear solver in the first nonlinear iteration is plotted in Fig. 3. It can be seen that the accuracy of the intermediate solvers affects the convergence rate of the linear solver. When the equations in the intermediate level are solved to a high precision, the convergence rate of the outer solver is steep. As one looses the tolerances for the intermediate solvers, the proposed algorithm requires more iterations for convergence. Yet, even for the tolerance as loose as $10^{-2}$, the convergence rate is still much steeper than that of the SIMPLE

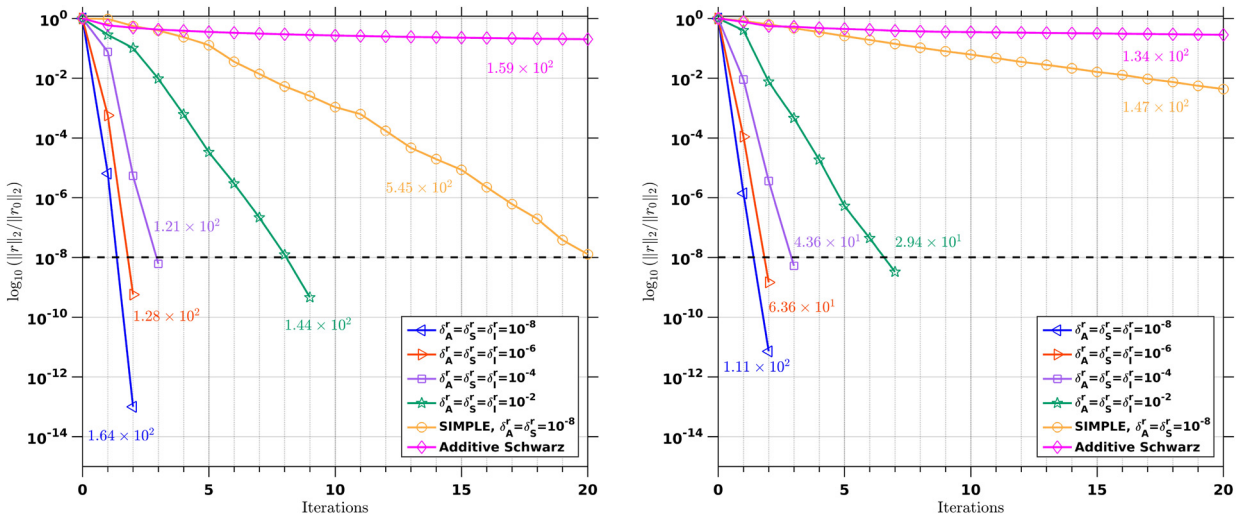


**Fig. 3.** Convergence history for $\Delta t = 10^{-1}$ (left) and $10^{-5}$ (right). The horizontal dashed black line indicates the prescribed stopping criterion for the relative error, which is $10^{-8}$ here. In the case of $\Delta t = 10^{-1}$, the SIMPLE method converges in 21 iterations, and the additive Schwarz method converges in 2644 iterations. In the case of $\Delta t = 10^{-5}$, the SIMPLE method converges in 71 iterations, and the additive Schwarz method converges in 2030 iterations. The numbers indicate the averaged time per nonlinear iteration in seconds.

**Table 3**
The performance of the linear solver with varying material properties. $\bar{n}$ represents the averaged number of FGMRES iterations; $\bar{n}_A$ represents the averaged number of iterations for solving with $\mathbf{A}$ in (4.3) and (4.5); $\bar{n}_S$ represents the averaged number of iterations for solving (4.4); $\bar{T}_L$ represents the averaged CPU time for one nonlinear iteration in seconds; $\nu$ represents the Poisson's ratio; $\eta$ is a non-dimensional scaling factor for the shear modulus.

| $\bar{n}$ [$\bar{n}_A$, $\bar{n}_S$] ($\bar{T}_L$) | $\eta = 10^{-2}$ | $\eta = 10^{0}$ | $\eta = 10^{2}$ |
|---|---|---|---|
| $\nu = 0.0$ | 2.0 [46.9, 15.9] (46.6) | 2.0 [48.1, 16.0] (48.3) | 2.0 [47.9, 15.3] (46.2) |
| $\nu = 0.1$ | 2.0 [48.5, 19.0] (49.2) | 2.0 [48.4, 17.9] (50.3) | 2.0 [48.1, 15.5] (46.5) |
| $\nu = 0.2$ | 2.0 [48.3, 20.2] (52.8) | 2.0 [48.0, 19.9] (52.9) | 2.0 [48.3, 16.8] (47.0) |
| $\nu = 0.3$ | 2.0 [47.9, 23.1] (56.4) | 2.0 [41.1, 21.5] (57.9) | 2.0 [48.5, 17.7] (48.6) |
| $\nu = 0.4$ | 2.0 [47.4, 28.5] (66.4) | 2.0 [48.2, 25.8] (65.5) | 2.0 [48.6, 19.2] (50.6) |
| $\nu = 0.5$ | 2.2 [47.1, 36.3] (101.2) | 2.0 [47.4, 24.6] (66.3) | 2.0 [46.5, 20.3] (48.6) |

preconditioner. The average time for solving the matrix problem per nonlinear iteration is reported in the figures as well. We observe that when choosing a strict tolerance for the intermediate and inner solvers, although convergence is achieved with fewer iterations, the cost per iteration is high and the overall time to solution is correspondingly high. A looser tolerance renders the application of the nested block preconditioner more cost-effective, and the overall algorithm is faster. In comparison with the SIMPLE and additive Schwarz methods, the proposed nested block preconditioning technique is fairly competitive.

*5.1.3. Performance with varying material properties*

In this example, we vary the material properties and study the robustness of the proposed preconditioner. The Poisson's ratio $\nu$ varies from 0.0 to 0.5, spanning the range relevant to most engineering and biological materials. The shear modulus $\mu$ is taken as $80.194 \times \eta$ MPa, wherein $\eta$ is a non-dimensional number. Correspondingly, the compression force is adjusted by multiplying with the scaling factor $\eta$ for values of $10^{-2}$, $10^{0}$, and $10^{2}$. The stopping condition for the linear solver is $\delta^r = 10^{-8}$, and we choose $\delta^r_A = \delta^r_S = \delta^r_I = 10^{-6}$. The mesh size is fixed to be $\Delta x = 1/480$, and the problem is simulated with 8 CPUs. The time step size is $\Delta t = 10^{-1}$ and we integrate the problem up to $T = 1.0$. We use a relatively large time step size here to make the matrix $\mathbf{A}$ dominated by the stiffness matrix. The statistics of the solver performance are collected over ten time steps (Table 3).

For all cases, the number of iterations for the outer solver maintains around two. In fact, it is only for the case of $\nu = 0.5$ and $\eta = 10^{-2}$ that the outer solver needs slightly more than two iterations. The number of iterations for solving with $\mathbf{A}$ in (4.3) and (4.5) is maintained around 47, and hence can be regarded as independent with respect to the material property. The number of iterations for solving (4.4) increases with increasing the Poisson's ratio. This can be explained by looking at $\mathbf{S} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$. The matrix $\mathbf{D}$ is dominated by the mass matrix scaled with a factor of $\beta$. As $\nu$ approaches 0.5, the isothermal compressibility coefficient $\beta$ goes to zero. Consequently, the well-conditioned matrix $\mathbf{D}$ diminishes, and the condition number of the Schur complement gets larger. This is reflected in the increase of $\bar{n}_S$ as $\nu$ goes from 0.0 to 0.5 for all three shear moduli. On the other hand, $\bar{n}_S$ increases as the material gets softer, and this trend is pronounced as

**Table 4**
The strong scaling performance. $\bar{n}$ represents the averaged number of FGMRES iterations for solving (3.16). $T_A$ and $T_L$ represent the timings for matrix assembly and linear solver, respectively. The efficiency is computed based on the total time.

| Proc. | $\bar{n}$ | $T_A$ (sec.) | $T_L$ (sec.) | Total (sec.) | Efficiency |
|---|---|---|---|---|---|
| 2 | 2.0 | $3.13 \times 10^3$ | $2.16 \times 10^4$ | $2.49 \times 10^4$ | 100% |
| 4 | 2.0 | $1.57 \times 10^3$ | $1.09 \times 10^4$ | $1.26 \times 10^4$ | 99% |
| 8 | 2.0 | $8.49 \times 10^2$ | $5.58 \times 10^3$ | $6.48 \times 10^3$ | 96% |
| 16 | 2.0 | $4.38 \times 10^2$ | $2.96 \times 10^3$ | $3.43 \times 10^3$ | 91% |
| 32 | 2.0 | $2.33 \times 10^2$ | $1.62 \times 10^3$ | $1.87 \times 10^3$ | 83% |
| 64 | 2.0 | $1.10 \times 10^2$ | $8.37 \times 10^2$ | $9.56 \times 10^2$ | 81% |
| 128 | 2.0 | $5.65 \times 10^1$ | $3.84 \times 10^2$ | $4.49 \times 10^2$ | 87% |

**Table 5**
Comparison of the averaged iteration counts and CPU time in seconds for the nested block preconditioner $\hat{\mathcal{P}}_{SCR}$, the SIMPLE preconditioner, and the additive Schwarz preconditioner. NC stands for no convergence. For the $\Delta x = 1/3840$ case, the additive Schwarz preconditioner failed to converge in 10000 iterations.

| $\frac{1}{\Delta x}$ | Proc. | $\hat{\mathcal{P}}_{SCR}$ | | | | SIMPLE | | Additive Schwarz | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\bar{n}$ | $\bar{n}_A$ | $\bar{n}_S$ | $\bar{T}_L$ | $\bar{n}$ | $\bar{T}_L$ | $\bar{n}$ | $\bar{T}_L$ |
| $\Delta t = 10^{-1}$ | | | | | | | | | |
| 480 | 8 | 2.3 | 31.7 | 6.7 | 19.7 | 13.3 | 21.6 | 1114.4 | 20.4 |
| 960 | 64 | 2.5 | 43.1 | 7.4 | 50.2 | 17.9 | 63.6 | 3368.9 | 106.8 |
| 1920 | 512 | 2.7 | 55.4 | 9.1 | 108.0 | 25.0 | 153.6 | 8642.4 | 305.4 |
| 3840 | 4096 | 2.9 | 68.8 | 9.6 | 220.8 | 47.6 | 504.2 | NC | NC |
| $\Delta t = 10^{-5}$ | | | | | | | | | |
| 480 | 8 | 2.3 | 4.6 | 16.1 | 5.3 | 22.7 | 6.3 | 916.0 | 17.0 |
| 960 | 64 | 2.0 | 6.9 | 26.4 | 18.5 | 38.6 | 31.6 | 2133.9 | 67.4 |
| 1920 | 512 | 2.0 | 9.1 | 34.3 | 52.8 | 65.7 | 71.0 | 9669.1 | 315.0 |
| 3840 | 4096 | 2.2 | 11.3 | 42.0 | 139.0 | 101.2 | 221.4 | NC | NC |

the Poisson's ratio gets larger. This can be explained by looking at $\mathbf{A}^{-1}$ in the Schur complement. For large time steps, $\mathbf{A}$ contains a significant contribution from the stiffness matrix, and the inverse of the stiffness matrix is proportional to $1/\mu$. It is known that diag $(\mathbf{A})$ is not a good candidate for approximating the stiffness matrix, and this is magnified for softer materials due to the factor $1/\mu$.

*5.1.4. Parallel performance*

We investigate the efficiency of the method by evaluating the fixed-size scalability performance. The spatial mesh size is $\Delta x = 1/1280$, with about $8.39 \times 10^6$ degrees of freedom. The time step size is fixed at $10^{-5}$, and we integrate the problem in time up to $T = 10^{-4}$. The stopping criterion for the FGMRES iteration is $\delta^r = 10^{-3}$; the tolerances for the intermediate and inner solvers are $\delta_A^r = \delta_S^r = \delta_I^r = 10^{-3}$. The communication speed of MPI messages between nodes is typically slower than that within a single node. To rule out the discrepancy of the communication speed, we run this test by assigning only one CPU per node. This means that the MPI messages are communicated purely by the cluster network. We observe that the efficiency of the numerical simulation is maintained at a high level (around 90%), and the average number of FGMRES iterations maintains at 2.0 for a wide range of processor counts (Table 4).

To compare the performance of different preconditioners, we also perform a weak scaling test of the solver, with $\delta^r = 10^{-3}$. Tolerances are set to $\delta_A^r = \delta_S^r = \delta_I^r = 10^{-3}$ for the nested block preconditioner and to $\delta_A^r = \delta_S^r = 10^{-3}$ for the SIMPLE preconditioner. The computational mesh is progressively refined and each CPU is assigned approximately $5.53 \times 10^4$ equations. We simulate the problem with two different time step sizes: $\Delta t = 10^{-1}$ and $10^{-5}$. The statistics of the solver performance are collected for ten time steps (Table 5). We observe that the iteration counts for the outer solver using the nested block preconditioner are independent of mesh refinement. At large time steps, $\mathbf{A}$ is dominated by the stiffness matrix and its solution procedure requires more iterations. In the meantime, the Schur complement has a better condition number and converges with fewer iterations. At small time steps, the situation is opposite. The matrix $\mathbf{A}$ is dominated by the mass matrix, and it can be solved with fewer iterations. The mesh refinement has an impact on the intermediate solvers, and we observe an increase of the number of iterations in $\bar{n}_A$ and $\bar{n}_S$. For the outer solver, $\bar{n}$ is maintained around a constant value, suggesting the outer solver is insensitive to the mesh refinement. The averaged CPU time $\bar{T}_L$ for the linear solver grows with mesh refinement, which is primarily attributed to the AMG preconditioner adopted. Indeed, there are known bottlenecks of the parallel AMG preconditioner [35,63], which prohibits ideal weak scalability of $\bar{T}_L$. For the SIMPLE preconditioner, the iteration counts and the CPU time grow faster than those of the nested block preconditioner. The additive Schwarz method converges faster per iteration. However, the number of iterations for convergence is much higher. For the finest mesh, the additive Schwarz method fails to converge in 10000 iterations. The proposed nested block preconditioner gives the most robust and efficient performance.

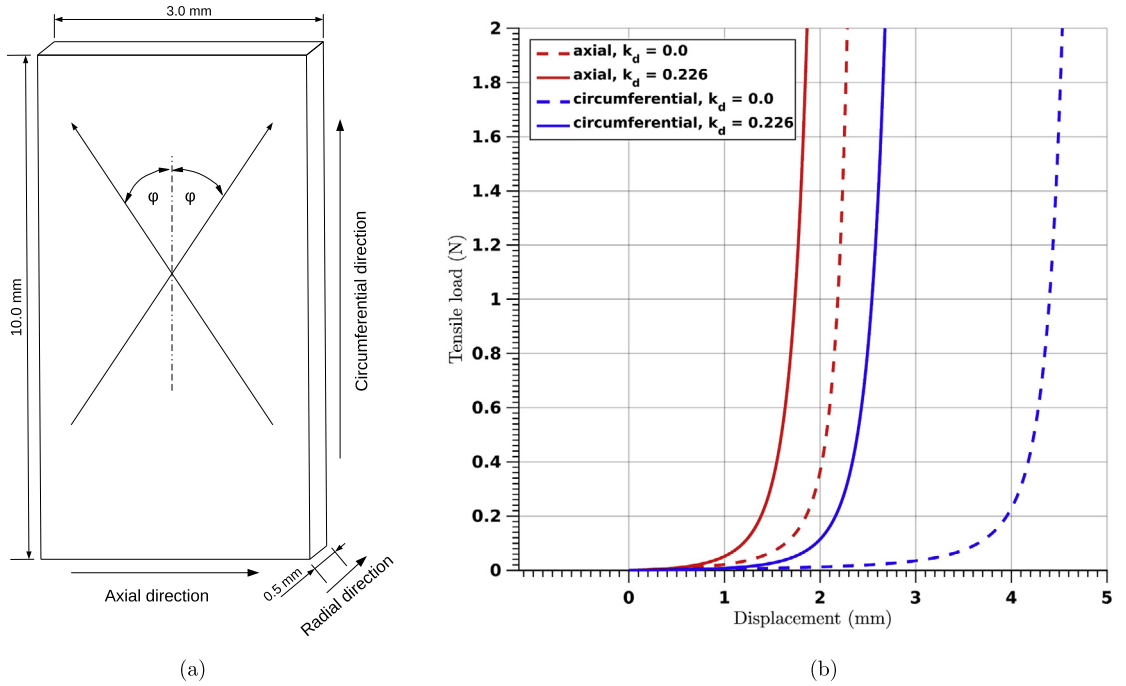


**Fig. 4.** Three-dimensional tensile test of an iliac adventitial strip: (a) geometry of the referential configuration; (b) computed load-displacement curves of the circumferential (blue) and axial specimens (red) with ($\kappa = 0.226$, solid curves) and without ($\kappa = 0.0$, dashed curves) dispersion of the collagen fibres.

### *5.2. Tensile test of an anisotropic fibre-reinforced hyperelastic soft tissue model*

In this example, we apply the proposed preconditioning technique to an anisotropic hyperelastic material model, which has been used to describe arterial tissue layers with distributed collagen fibres. The isochoric and volumetric parts of the free energy are

$$G_{ich}(\tilde{\boldsymbol{C}}) = G^{g}_{ich}(\tilde{\boldsymbol{C}}) + \sum_{i=1,2} G^{f_i}_{ich}(\tilde{\boldsymbol{C}}), \quad G_{vol}(p) = \frac{p}{\rho_0},$$

$$G^{g}_{ich}(\tilde{\boldsymbol{C}}) = \frac{\mu}{2\rho_0}\left(\mathrm{tr}\tilde{\boldsymbol{C}} - 3\right), \quad G^{f_i}_{ich}(\tilde{\boldsymbol{C}}) = \frac{k_1}{2k_2\rho_0}\left(e^{k_2\bar{E}_i^2} - 1\right),$$

$$\bar{E}_i := \boldsymbol{H}_i : \tilde{\boldsymbol{C}} - 1, \quad \boldsymbol{H}_i := k_d\boldsymbol{I} + (1 - 3k_d)(\boldsymbol{a}_i \otimes \boldsymbol{a}_i).$$

In the above, $G^{g}_{ich}$ models the groundmatrix via an isotropic Neo-Hookean material, with $\mu$ being the shear modulus; $G^{f_i}_{ich}$ models the $i$th family of collagen fibres by an exponential function. In $G^{f_i}_{ich}$, $\boldsymbol{a}_i$ is a unit vector that describes the mean orientation of the $i$th family of fibres in the reference configuration. The parameter $k_d \in [0, 1/3]$ is a structural parameter that characterizes the dispersion of the collagen fibres. For ideally aligned fibres, the dispersion parameter $k_d$ is 0, while for isotropically distributed fibres, it takes the value 1/3. The parameter $k_1$ is a material parameter that describes the stiffness of the fibre, and $k_2$ is a non-dimensional parameter. The volumetric energy $G_{vol}$ indicates that the model is fully incompressible. Interested readers are referred to [37] for detailed discussions of the histology and constitutive modeling of the arterial layers. In the numerical study, we perform a tensile test for the tissue model. Following [37], the geometry of the specimen has length 10.0 mm, width 3.0 mm, and thickness 0.5 mm. The material parameters are $\mu = 7.64$ kPa, $k_1 = 996.6$ kPa, $k_2 = 524.6$. Assuming that the fibre orientation has no radial component, the unit vector is characterized completely by $\varphi$, the angle between the circumferential direction and the mean fibre orientation direction (see Fig. 4 (a)). For the circumferential specimen, $\varphi = 49.98°$; for the axial specimen, $\varphi = 40.02°$. On the loading surface, traction force is applied and the face is constrained to move only in the loading direction. Symmetry boundary conditions are properly applied, and we only consider one-eighth of the specimen in the simulations.

Before studying the solver performance, we perform a simulation with 3.5 million unstructured linear tetrahedral elements to examine the VMS formulation for this material model. In this study, the tensile test is performed in a dynamic approach. The loading force is applied as a linear function of time and reaches 2 N in 100 seconds. We set the density of the tissue as 1.0 g/cm$^3$. The tensile load-displacement curves for the circumferential and axial specimens with $k_d = 0.0$ and 0.226 are plotted in Fig. 4 (b). We observe that before the fibres align along the loading direction, the groundmatrix

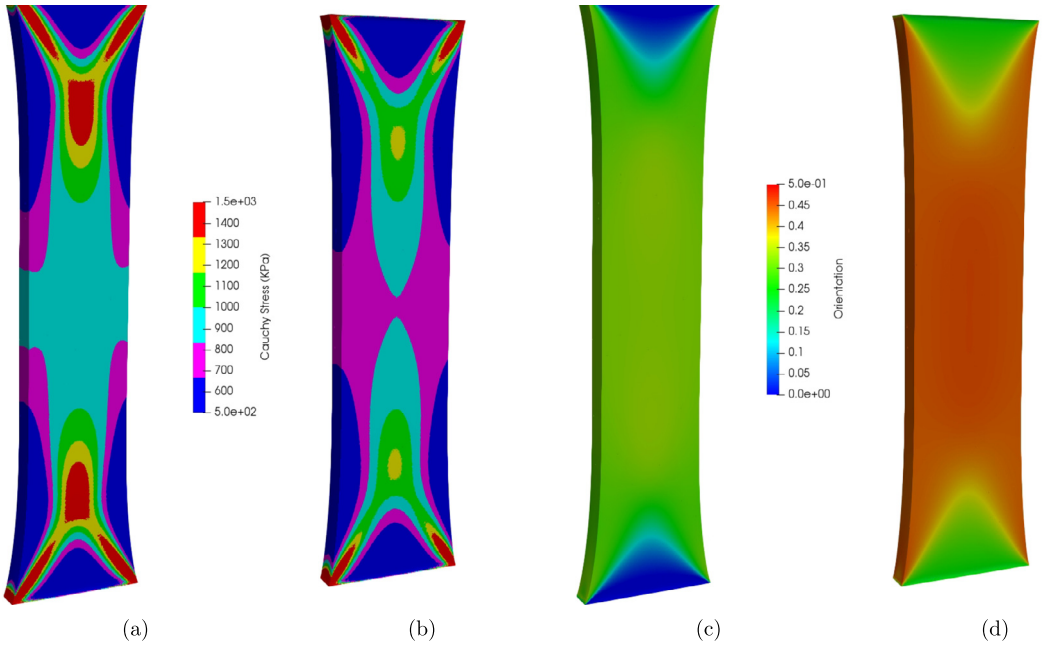


**Fig. 5.** Three-dimensional tensile test: Cauchy stress in the loading direction are plotted for the circumferential (a) and axial (b) specimens. The mean orientation of the collagen fibres in the current configuration are plotted for the circumferential (c) and axial (d) specimens.

provides the load carry capacity and the material response is very soft. When the fibres rotate to align with the loading direction, they take over the load burden, the material becomes stiffer, and the stiffness grows exponentially. For the axial specimen, the mean orientation of the fibres are closer to the loading direction, and hence it stiffens earlier than the circumferential specimen. Compared with the dispersed case, the specimen with perfectly aligned fibres (i.e., $k_d = 0.0$) needs a significant amount of rotation before they can carry load. In Fig. 5 (a) and (b), the Cauchy stresses in the tensile direction for the circumferential and axial specimens with $k_d = 0.226$ at the tensile load 1.0 N are illustrated. The value of $\boldsymbol{a}_1 \cdot \boldsymbol{C}\boldsymbol{a}_2 / \|\boldsymbol{F}\boldsymbol{a}_1\| \|\boldsymbol{F}\boldsymbol{a}_2\|$ characterizes the current fibre alignment, and it is illustrated in Fig. 5 (c) and (d) for the circumferential and axial specimens. The maximum values in these specimens are 0.347 and 0.459, respectively. Correspondingly, the angles between the current mean fibre direction and the circumferential direction are 34.85° and 31.34°, respectively. In the following discussion, the problem has been non-dimensionalized by the centimetre-gram-second units. Except the study performed in Section 5.2.3, we adopt the axial specimen with the dispersion parameter $k_d = 0.226$ as the model problem for the study of the solver performance.

*5.2.1. Performance with varying inner solver accuracy*

In this test, we study the impact of the inner solver accuracy on the iterative solution algorithm. We fix the mesh size to be 1/400 and the time step size to be $10^{-5}$. The simulation is performed with 8 CPUs. In this test, the settings of the linear solver are identical to the study performed in Section 5.1.2. The statistics of the solver are collected for the first time step of the simulation with varying values of $\delta_I^r$ (Table 6). The impact of the inner solver accuracy is similar to what was observed in Section 5.1.1. It is confirmed that using the inner solver may significantly improve the convergence rate of the linear solver. In both cases, the optimal performance in terms of time to solution is achieved by setting $\delta_I^r = 10^2 \delta_S^r$.

*5.2.2. Performance with varying intermediate solver accuracy*

We examine the solver performance for anisotropic hyperelastic materials with varying tolerances for the intermediate solvers. The mesh size is fixed to be $\Delta x = 1/400$, and the time step sizes are fixed to be $\Delta t = 10^{-1}$ and $10^{-5}$. The simulations are performed with 8 CPUs. We choose $\delta^r = 10^{-8}$ and vary the values of $\delta_A^r = \delta_S^r = \delta_I^r$ from $10^{-8}$ to $10^{-2}$. The SIMPLE preconditioner and the additive Schwarz preconditioner are also simulated for comparison. In the SIMPLE preconditioner, the block matrices $\mathbf{A}$ and $\hat{\mathbf{S}}$ are solved with $\delta_A^r = \delta_S^r = 10^{-8}$. The convergence history of the linear solver in the first nonlinear iteration is plotted in Fig. 6. We observe that the nested block preconditioner performs robustly with a strict choice of the intermediate and inner solver tolerances. When the tolerances for the intermediate and inner solvers are loose ($10^{-2}$) and the time step is large ($\Delta t = 10^{-1}$), the convergence rate of the nested block preconditioner slows dramatically and is slower than the SIMPLE preconditioner. It should be emphasized that the SIMPLE preconditioner uses a very strict tolerance ($\delta_A^r = \delta_S^r = 10^{-8}$) here. We also note that the additive Schwarz preconditioner fails to converge to the prescribed tolerance in 10000 iterations when $\Delta t = 10^{-1}$.

**Table 6**
The impact of the accuracy of the inner solver on the performance of the linear solver. The CPU time is collected for the linear solver only; $\hat{l}$ represents the total number of nonlinear iterations; $n$ represents the total number of FGMRES iterations; $\bar{n}_A$ represents the averaged number of iterations for solving with **A** in (4.3) and (4.5); $\bar{n}_S$ represents the averaged number of iterations for solving (4.4); $\bar{n}_I$ represents the averaged number of iterations for solving (4.6).

| | $\delta_I^r$ | CPU time (sec.) | $\hat{l}$ | $n$ | $\bar{n}_A$ | $\bar{n}_S$ | $\bar{n}_I$ |
|---|---|---|---|---|---|---|---|
| $\delta_A^r = \delta_S^r = 10^{-10}$ | $10^0$ | $7.56 \times 10^1$ | 1 | 47 | 16.81 | 19.81 | – |
| | $10^{-2}$ | $7.19 \times 10^1$ | 1 | 9 | 15.78 | 58.56 | 1.95 |
| | $10^{-4}$ | $6.52 \times 10^1$ | 1 | 5 | 15.30 | 58.80 | 3.75 |
| | $10^{-6}$ | $6.20 \times 10^1$ | 1 | 3 | 14.33 | 58.33 | 7.81 |
| | $10^{-8}$ | $5.83 \times 10^1$ | 1 | 2 | 13.75 | 59.50 | 11.85 |
| | $10^{-10}$ | $7.55 \times 10^1$ | 1 | 2 | 13.75 | 60.00 | 15.19 |
| $\delta_A^r = \delta_S^r = 10^{-6}$ | $10^0$ | $4.38 \times 10^1$ | 1 | 47 | 7.83 | 12.02 | – |
| | $10^{-2}$ | $4.20 \times 10^1$ | 1 | 9 | 8.17 | 34.22 | 1.97 |
| | $10^{-4}$ | $3.67 \times 10^1$ | 1 | 5 | 7.90 | 34.40 | 3.40 |
| | $10^{-6}$ | $4.73 \times 10^1$ | 1 | 4 | 7.50 | 35.75 | 7.37 |
| | $10^{-8}$ | $6.87 \times 10^1$ | 1 | 4 | 7.00 | 35.75 | 11.50 |
| | $10^{-10}$ | $8.23 \times 10^1$ | 1 | 4 | 7.00 | 35.75 | 15.47 |

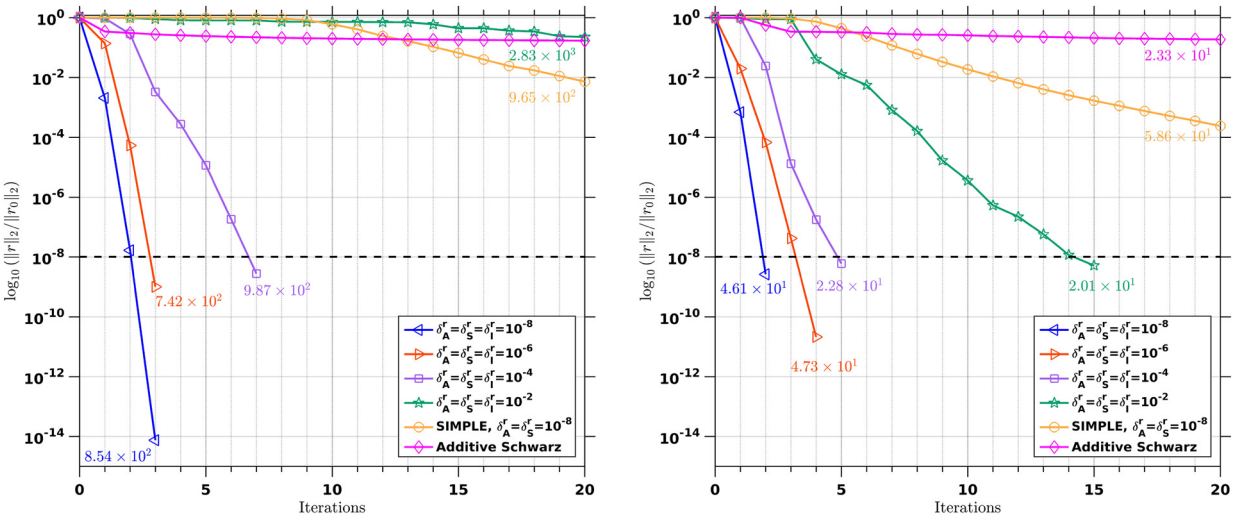


**Fig. 6.** Convergence history for $\Delta t = 10^{-1}$ (left) and $10^{-5}$ (right). The horizontal dashed black line indicates the prescribed stopping criterion for the relative error, which is $10^{-8}$ here. In the case of $\Delta t = 10^{-1}$, the block preconditioner with tolerance $10^{-2}$ converge in 90 iterations, the SIMPLE method converges in 45 steps, and the additive Schwarz method failed to converge. In the case of $\Delta t = 10^{-5}$, the SIMPLE method converges in 46 iterations, and the additive Schwarz method converges in 1070 iterations. The numbers indicate the averaged time per nonlinear iteration in seconds.

**Table 7**
The performance of the nested block preconditioner with varying fibre orientations and dispersions. $\bar{n}$ represents the averaged number of FGMRES iterations; $\bar{n}_A$ represents the averaged number of iterations for solving with **A** in (4.3) and (4.5); $\bar{n}_S$ represents the averaged number of iterations for solving (4.4); $\bar{T}_L$ represents the averaged CPU time for one nonlinear iteration in seconds; $\varphi$ is the collagen fibre mean orientation; $k_d$ is the dispersion parameter.

| $\bar{n}$ [$\bar{n}_A$, $\bar{n}_S$] ($\bar{T}_L$) | $k_d = 0.1$ | $k_d = 0.2$ | $k_d = 0.3$ |
|---|---|---|---|
| $\varphi = 20°$ | 3.0 [214.1, 17.6] ($4.8 \times 10^1$) | 3.0 [161.2, 18.0] ($3.4 \times 10^1$) | 3.0 [103.4, 19.7] ($2.1 \times 10^1$) |
| $\varphi = 40°$ | 3.0 [241.3, 17.7] ($5.8 \times 10^1$) | 3.0 [176.8, 18.4] ($4.1 \times 10^1$) | 2.9 [105.7, 20.5] ($2.1 \times 10^1$) |
| $\varphi = 60°$ | 2.8 [221.4, 17.8] ($4.6 \times 10^1$) | 2.9 [169.1, 19.2] ($3.7 \times 10^1$) | 2.9 [104.5, 20.9] ($2.1 \times 10^1$) |
| $\varphi = 80°$ | 2.9 [220.8, 18.1] ($5.3 \times 10^1$) | 3.0 [168.2, 19.8] ($4.1 \times 10^1$) | 3.0 [103.0, 20.9] ($2.2 \times 10^1$) |

### *5.2.3. Performance with varying fibre orientations and dispersions*

In this test, we examine the robustness of the solver with different collagen fibre orientations and dispersions. The structure of the arterial wall is described by the collagen fibre mean orientation $\varphi$ and the dispersion parameter $k_d$. We vary the value of $\varphi$ from 20° to 80°, and the value of $k_d$ from 0.1 to 0.3. The rest material properties are kept the same as the ones used in the previous studies. The simulations are performed with $\Delta x = 1/100$ on 8 CPUs. The time step size is $\Delta t = 10^{-1}$, and we simulate the problem up to $T = 1.0$ to collect statistics of the solver performance. The stopping condition for the FGMRES iteration is $\delta^r = 10^{-8}$, and we choose $\delta_A^r = \delta_S^r = \delta_I^r = 10^{-6}$. The averaged number of iterations and the averaged CPU time for one nonlinear iteration is reported in Table 7.

**Table 8**
Comparison of the averaged iteration counts and CPU time in seconds for the nested block preconditioner $\hat{\mathcal{P}}_{SCR}$, the SIMPLE preconditioner, and the additive Schwarz preconditioner. NC stands for no convergence. For the $\Delta t = 10^{-1}$ case, the additive Schwarz preconditioner failed to achieve convergence in 10000 iterations.

| $\frac{1}{\Delta x}$ | Proc. | $\hat{\mathcal{P}}_{SCR}$ | | | | SIMPLE | | Additive Schwarz | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\bar{n}$ | $\bar{n}_A$ | $\bar{n}_S$ | $\bar{T}_L$ | $\bar{n}$ | $\bar{T}_L$ | $\bar{n}$ | $\bar{T}_L$ |
| $\Delta t = 10^{-1}$ | | | | | | | | | |
| 200 | 8 | 6.2 | 207.1 | 10.3 | 579.3 | 54.1 | 888.5 | NC | NC |
| 400 | 64 | 7.8 | 331.3 | 10.5 | 2062.7 | 102.5 | 6262.8 | NC | NC |
| 600 | 216 | 10.3 | 389.4 | 11.1 | 3204.1 | 140.5 | 8051.3 | NC | NC |
| $\Delta t = 10^{-5}$ | | | | | | | | | |
| 200 | 8 | 4.0 | 2.3 | 15.5 | 9.7 | 22.6 | 9.4 | 485.6 | 10.8 |
| 400 | 64 | 4.1 | 3.1 | 18.6 | 29.2 | 45.3 | 53.3 | 986.3 | 47.76 |
| 600 | 216 | 5.9 | 3.9 | 20.3 | 119.8 | 71.3 | 202.5 | 1453.0 | 431.6 |

We observe that the outer solver converges in around three iterations regardless of the structural properties. In the intermediate level, the linear solver for **S** is not sensitive to the two structural parameters; the linear solver for **A** is affected by both parameters. The dispersion parameter $k_d$ has a significant impact on the performance of the solver associated with **A**. For the case of $k_d = 0.1$, the solver for **A** requires slightly more than 200 iterations for convergence; for the case of $k_d = 0.3$, the number of iterations drops to around 100. As the dispersion parameter grows, there are more fibres providing stiffness. Thus, the trend of $\bar{n}_A$ is in agreement with the observations made in Section 5.1.3.

*5.2.4. Parallel performance*

We compare the performance of different preconditioners by performing a weak scaling test. The tolerance for the linear solver is set to be $\delta^r = 10^{-3}$. In the nested block preconditioner, we set $\delta^r_A = \delta^r_S = \delta^r_I = 10^{-3}$, and we use $\delta^r_A = \delta^r_S = 10^{-3}$ for the SIMPLE preconditioner. The computational mesh is progressively refined and each CPU is assigned with approximately $6.0 \times 10^4$ equations. We simulate the problem with two different time step sizes: $\Delta t = 10^{-1}$ and $10^{-5}$. The statistics of the solver performance are collected for five time steps, and the results are reported in Table 8. The number of iterations at the intermediate level shows a similar trend to the isotropic case studied in Section 5.1.4. The difference is that, for the anisotropic material, the solver for **A** requires more iterations to converge when the time step size is large. The degradation of the AMG preconditioner for anisotropic problems is known, and using a higher complexity coarsening, like the Falgout method, will improve the performance [57]. Notably, for large time steps, the additive Schwarz preconditioner just cannot deliver converged solutions within 10000 iterations, regardless of the spatial mesh size. Examining the results, the proposed nested block preconditioner gives the most robust and efficient performance for most of the cases considered.

## 6. Conclusions

In this work, we designed a preconditioning technique based the novel hyper-elastodynamics formulation [1]. This preconditioning technique is based on a series of block factorizations in the Newton–Raphson solution procedure [1,3,44] and is inspired from the preconditioning techniques developed in the CFD community [27–30]. It uses the Schur complement reduction with relaxed tolerances as the preconditioner inside a Krylov subspace method. This strategy enjoys the merits of both the SCR approach and the fully coupled approaches. It shows better robustness and efficiency in comparison with the SIMPLE and the additive Schwarz preconditioners. Tuning the intermediate and the inner solvers allows the user to adjust the nested algorithm for specific problems to attain a balance between robustness and efficiency. In this work, to make the presentation coherent, we adopted the same solver at the intermediate and the inner levels. In practice, one is advised to flexibly apply the most efficient solver at the inner level. For example, one may symmetrize the matrix in (4.6) [30] and use the conjugate gradient method as the inner solver. In our experience, this will further reduce the computational cost. In all, the methodology developed in this work provides a sound basis for the design of effective preconditioning techniques for hyper-elastodynamics.

There are several promising directions for future work. (1) Improvements will be made to design a better preconditioner for the Schur complement. It is tempting to consider using the sparse approximate inverse method to construct this preconditioner [64]. (2) Geometric multigrid preconditioners will be developed to replace the AMG preconditioner. This is expected to further improve the scalability of the proposed solution method. (3) This preconditioning technique will be extended to inelastic calculations [39,40] as well as FSI problems [1].

## Acknowledgements

This work is supported by the National Institutes of Health under the award numbers 1R01HL121754 and 1R01HL123689, the National Science Foundation (NSF) CAREER award OCI-1150184, and computational resources from the Extreme Science and Engineering Discovery Environment (XSEDE) supported by the NSF grant ACI-1053575. The authors acknowledge TACC

at the University of Texas at Austin for providing computing resources that have contributed to the research results reported within this paper.

## Appendix A. Consistent linearization

We report the explicit formulas of the residual vectors and tangent matrices used in the Newton–Raphson solution procedure at the iteration step $l$. For notational simplicity, the subscript $(l)$ is neglected in the following discussion.

$$\mathbf{R}_m = \left[\mathrm{R}^i_{m,A}\right], \tag{A.1}$$

$$\mathrm{R}^i_{m,A} = \int_{\Omega_{\boldsymbol{X}}} N_A J\rho\dot{v}_i + N_{A,I}\tilde{\boldsymbol{P}}_{iI} - N_{A,i}Jp - N_A J\rho b_i d\Omega_{\boldsymbol{X}} - \int_{\Gamma^H_{\boldsymbol{X}}} N_A H_i d\Gamma_{\boldsymbol{X}}, \tag{A.2}$$

$$\mathbf{R}_p = \left[\mathrm{R}_{p,A}\right], \tag{A.3}$$

$$\mathrm{R}_{p,A} = \int_{\Omega_{\boldsymbol{X}}} JN_A\left(\beta\dot{p} + v_{i,i}\right)d\Omega_{\boldsymbol{X}} + \sum_e \int_{\Omega^e_{\boldsymbol{X}}} \tau^e_M N_{A,i}\left(\rho J\dot{v}_i - \tilde{\boldsymbol{P}}_{iJ,J} + Jp_{,i} - \rho Jb_i\right)d\Omega_{\boldsymbol{X}}. \tag{A.4}$$

In the above, we used the following notation conventions,

$$N_{A,I} := \frac{\partial N_A}{\partial X_I}, \quad N_{A,i} := \frac{\partial N_A}{\partial x_i} = \frac{\partial N_A}{\partial X_I}\frac{\partial X_I}{\partial x_i} = \frac{\partial N_A}{\partial X_I}F^{-1}_{Ii}, \quad p_{,i} = p_{,I}F^{-1}_{Ii}, \quad \tilde{\boldsymbol{P}}_{iI} := J\boldsymbol{\sigma}^{dev}_{ij}\boldsymbol{F}^{-1}_{Ij}.$$

Note that $\rho = \rho(p)$ and $\beta = \beta(p)$ are given by the constitutive relations, and $H_i := h_i \circ \boldsymbol{\varphi}_t$.

$$\mathbf{A} = \left[\mathrm{A}^{ij}_{AB}\right], \tag{A.5}$$

$$\begin{aligned}
\mathrm{A}^{ij}_{AB} =& \alpha_m \int_{\Omega_{\boldsymbol{X}}} J\rho N_A N_B d\Omega_{\boldsymbol{X}}\delta_{ij} + \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\int_{\Omega_{\boldsymbol{X}}} N_{A,I}\left(\tilde{\boldsymbol{S}}_{IJ}\delta_{ij} + \mathbb{A}^{ich}_{iIjJ}\right)N_{B,J}d\Omega_{\boldsymbol{X}} \\
&+ \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\int_{\Omega_{\boldsymbol{X}}} JpN_{A,I}\left(F^{-1}_{Ij}F^{-1}_{Ji} - F^{-1}_{Ii}F^{-1}_{Jj}\right)N_{B,J}d\Omega_{\boldsymbol{X}} \\
&+ \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\int_{\Omega_{\boldsymbol{X}}} N_A\rho J\left(\dot{v}_i - b_i\right)N_{B,j}d\Omega_{\boldsymbol{X}},
\end{aligned}$$

$$\mathbf{B} = \left[\mathrm{B}^i_{AB}\right], \tag{A.6}$$

$$\mathrm{B}^i_{AB} = \alpha_f\gamma\Delta t\int_{\Omega_{\boldsymbol{X}}} \rho_{,p}J(\dot{v}_i - b_i)N_AN_B - JN_{A,i}N_Bd\Omega_{\boldsymbol{X}},$$

$$\mathbf{C} = \left[\mathrm{C}^j_{AB}\right], \tag{A.7}$$

$$\begin{aligned}
\mathrm{C}^j_{AB} =& \alpha_m\sum_e\int_{\Omega^e_{\boldsymbol{X}}}\tau^e_M\rho JN_{A,j}N_Bd\Omega_{\boldsymbol{X}} + \alpha_f\gamma\Delta t\int_{\Omega_{\boldsymbol{X}}} JN_AN_{B,j}d\Omega_{\boldsymbol{X}} \\
&+ \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\int_{\Omega_{\boldsymbol{X}}} J\beta\dot{p}N_AN_{B,j} + JN_A\left(v_{i,i}N_{B,j} - v_{i,j}N_{B,i}\right)d\Omega_{\boldsymbol{X}} \\
&- \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\sum_e\int_{\Omega^e_{\boldsymbol{X}}}\tau^e_M N_{A,j}N_{B,i}\left(\rho J\dot{v}_i - \tilde{\boldsymbol{P}}_{iI,I} + Jp_{,i} - \rho Jb_i\right)d\Omega_{\boldsymbol{X}} \\
&+ \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\sum_e\int_{\Omega^e_{\boldsymbol{X}}}\tau^e_M JN_{A,i}\left(p_{,i}N_{B,j} - p_{,j}N_{B,i}\right)d\Omega_{\boldsymbol{X}} \\
&+ \frac{\left(\alpha_f\gamma\Delta t_n\right)^2}{\alpha_m}\sum_e\int_{\Omega^e_{\boldsymbol{X}}}\tau^e_M N_{A,i}N_{B,j}\rho J\left(\dot{v}_i - b_i\right)d\Omega_{\boldsymbol{X}}
\end{aligned}$$

$$- \frac{\left(\alpha_f \gamma \Delta t_n\right)^2}{\alpha_m} \sum_e \int_{\Omega^e_{\boldsymbol{X}}} \tau^e_M N_{A,i} \left( \tilde{\boldsymbol{S}}_{MN} \delta_{ij} + \mathbb{A}^{ich}_{iMjN} \right) N_{B,MN} d\Omega_{\boldsymbol{X}}, \tag{A.8}$$

$$\mathbf{D} = [\mathrm{D}_{AB}], \tag{A.9}$$

$$\begin{aligned} \mathrm{D}_{AB} =& \alpha_m \int_{\Omega_{\boldsymbol{X}}} J \beta N_A N_B d\Omega_{\boldsymbol{X}} + \alpha_f \gamma \Delta t \int_{\Omega_{\boldsymbol{X}}} J \beta_{,p} \dot{p} N_A N_B d\Omega_{\boldsymbol{X}} \\ &+ \alpha_f \gamma \Delta t \sum_e \int_{\Omega^e_{\boldsymbol{X}}} J \tau^e_M \left( N_{A,i} N_{B,i} + \rho_{,p} \left( \dot{v}_i - b_i \right) N_{A,i} N_B \right) d\Omega_{\boldsymbol{X}}. \end{aligned} \tag{A.10}$$

In $\mathrm{A}^{ij}_{AB}$ and $\mathrm{C}^{j}_{AB}$, we used the following notation,

$$\mathbb{A}^{ich}_{iIjJ} := \frac{\partial G_{ich}}{\partial F_{iI} \partial F_{jJ}}.$$

## References


[1] J. Liu, A. Marsden, A unified continuum and variational multiscale formulation for fluids, solids, and fluid-structure interaction, Comput. Methods Appl. Mech. Eng. 337 (2018) 549–597.
[2] T. Hughes, Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods, Comput. Methods Appl. Mech. Eng. 127 (1995) 387–401.
[3] G. Scovazzi, B. Carnes, X. Zeng, S. Rossi, A simple, stable, and accurate linear tetrahedral finite element for transient, nearly, and fully incompressible solid dynamics: a dynamic variational multiscale approach, Int. J. Numer. Methods Eng. 106 (2016) 799–839.
[4] A. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comput. 22 (1968) 745–762.
[5] R. Teman, Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionnaires (II), Arch. Ration. Mech. Anal. 33 (1969) 377–385.
[6] M. Benzi, G. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137.
[7] H. Elman, D. Silvester, A. Wathen, Finite Elements and Fast Iterative Solvers, 2nd edition, Oxford University Press, 2014.
[8] S. Turek, Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach, Springer Science & Business Media, 1999.
[9] D. Keyes, et al., Multiphysics simulations: challenges and opportunities, Int. J. High Perform. Comput. Appl. 27 (2013) 4–83.
[10] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[11] J. van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, SIAM J. Sci. Stat. Comput. 7 (1986) 870–891.
[12] G. Karniadakis, M. Israeli, S. Orszag, High-order splitting methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 97 (1991) 414–443.
[13] J. Guermond, J. Shen, Velocity-correction projection methods for incompressible flows, SIAM J. Numer. Anal. 41 (2003) 112–134.
[14] J. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Eng. 195 (2006) 6011–6045.
[15] J. Perot, An analysis of the fractional step method, J. Comput. Phys. 108 (1993) 51–58.
[16] A. Quarteroni, F. Saleri, A. Veneziani, Factorization methods for the numerical approximation of Navier–Stokes equations, Comput. Methods Appl. Mech. Eng. 188.
[17] 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.
[18] 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.
[19] D. Silvester, A. Wathen, Fast iterative solution of stabilised Stokes systems Part II: using general block preconditioners, SIAM J. Numer. Anal. 31 (1994) 1352–1367.
[20] H. Elman, Preconditioning for the steady-state Navier–Stokes equations with low viscosity, SIAM J. Sci. Comput. 20 (1999) 1299–1316.
[21] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state Navier–Stokes equations, SIAM J. Sci. Comput. 24 (2002) 237–256.
[22] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, Block preconditioners based on approximate commutators, SIAM J. Sci. Comput. 27 (2006) 1651–1668.
[23] 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.
[24] D. May, L. Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, Phys. Earth Planet. Inter. 171 (2008) 33–47.
[25] L. Lun, A. Yeckel, J. Derby, A Schur complement formulation for solving free-boundary, Stefan problems of phase change, J. Comput. Phys. 229 (2010) 7942–7955.
[26] M. Furuichi, D. May, P. Tackley, Development of a Stokes flow solver robust to large viscosity jumps using a Schur complement approach with mixed precision arithmetic, J. Comput. Phys. 230 (2011) 8835–8851.
[27] R. Bank, B. Welfert, H. Yserentant, A class of iterative methods for solving saddle point problems, Numer. Math. 56 (1990) 645–666.
[28] A. Baggag, A. Sameh, A nested iterative scheme for indefinite linear systems in particulate flows, Comput. Methods Appl. Mech. Eng. 193 (2004) 1923–1957.
[29] 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.
[30] 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.
[31] M. Moghadam, Y. Bazilevs, A. Marsden, A bi-partitioned iterative algorithm for solving linear systems obtained from incompressible flow problems, Comput. Methods Appl. Mech. Eng. 286 (2015) 40–62.
[32] E. Cyr, J. Shadid, R. Tuminaro, Stabilization and scalable block preconditioning for the Navier–Stokes equations, J. Comput. Phys. 231 (2012) 345–363.
[33] Y. Saad, M. Schultz GMRES, A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.

[34] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (1993) 461–469.
[35] U. Yang, BoomerAMG: a parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math. 41 (2002) 155–177.
[36] S. Reese, P. Wriggers, B. Reddy, A new locking-free brick element technique for large deformation problems in elasticity, Comput. Struct. 75 (2000) 291–304.
[37] T. Gasser, R. Ogden, G. Holzapfel, Hyperelastic modelling of arterial layers with distributed collagen fibre orientations, J. R. Soc. Interface 3 (2006) 15–35.
[38] K. Jansen, C. Whiting, G. Hulbert, A generalized-$\alpha$ method for integrating the filtered Navier–Stokes equations with a stabilized finite element method, Comput. Methods Appl. Mech. Eng. 190 (2000) 305–319.
[39] X. Zeng, G. Scovazzi, N. Abboud, O. Colomés Gene, S. Rossi, A dynamic variational multiscale method for viscoelasticity using linear tetrahedral elements, Int. J. Numer. Methods Eng. 112 (2017) 1951–2003.
[40] N. Abboud, G. Scovazzi, Elastoplasticity with linear tetrahedral elements: a variational multiscale method, Int. J. Numer. Methods Eng. 115 (2018) 913–955.
[41] T. Hughes, G. Hulbert, Space-time finite element methods for elastodynamics: formulation and error estimates, Comput. Methods Appl. Mech. Eng. 66 (1988) 339–363.
[42] J. Chung, G. Hulbert, A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-$\alpha$ method, J. Appl. Mech. 60 (1993) 371–375.
[43] C. Kadapa, W. Dettmer, D. Perić, On the advantages of using the first-order generalised-alpha scheme for structural dynamic problems, Comput. Struct. 193 (2017) 226–238.
[44] S. Rossi, N. Abboud, G. Scovazzi, Implicit finite incompressible elastodynamics with linear finite elements: a stabilized method in rate form, Comput. Methods Appl. Mech. Eng. 311 (2016) 208–249.
[45] F. Shakib, T. Hughes, Z. Johan, A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis, Comput. Methods Appl. Mech. Eng. 75 (1989) 415–456.
[46] L. Berger-Vergiat, C. McAuliffe, H. Waisman, Parallel preconditioners for monolithic solution of shear bands, J. Comput. Phys. 304 (2016) 359–379.
[47] 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.
[48] 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.
[49] F. Verdugo, W. Wall, Unified computational framework for the efficient solution of n-field coupled problems with monolithic schemes, Comput. Methods Appl. Mech. Eng. 310 (2016) 335–366.
[50] J. White, R. Borja, Block-preconditioned Newton-Krylov solvers for fully coupled flow and geomechanics, Comput. Geosci. 15 (2011) 647.
[51] A. Wathen, D. Silvester, Fast iterative solution of stabilised Stokes systems. Part I: Using simple diagonal preconditioners, SIAM J. Numer. Anal. 30 (1993) 630–649.
[52] H. Elman, D. Silvester, A. Wathen, Block preconditioners for the discrete incompressible Navier–Stokes equations, Int. J. Numer. Methods Fluids 40 (2002) 333–344.
[53] I. Ipsen, A note on preconditioning nonsymmetric matrices, SIAM J. Sci. Comput. 23 (2001) 1050–1051.
[54] M. Murphy, G. Golub, A. Wathen, A note on preconditioning for indefinite linear systems, SIAM J. Sci. Comput. 21 (2000) 1969–1972.
[55] A.E. Maliki, M. Fortin, N. Tardieu, A. Fortin, Iterative solvers for 3D linear and nonlinear elasticity problems: displacement and mixed formulations, Int. J. Numer. Methods Eng. 83 (2010) 1780–1802.
[56] V. Gurev, P. Pathmanathan, J. Fattebert, H. Wen, J. Magerlein, R. Gray, D. Richards, J. Rice, A high-resolution computational model of the deforming human heart, Biomech. Model. Mechanobiol. 14 (2015) 829–849.
[57] V. Henson, U. Yang, BoomerAMG: a parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math. 41 (2002) 155–177.
[58] R. Falgout, U. Yang, hypre: a library of high performance preconditioners, in: International Conference on Computational Science, Springer, 2002, pp. 632–641.
[59] B. Smith, P. Bjorstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 2004.
[60] 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.
[61] 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.
[62] A. Bouras, V. Frayssé, Inexact matrix–vector products in Krylov methods for solving linear systems: a relaxation strategy, SIAM J. Matrix Anal. Appl. 26 (2005) 660–678.
[63] H.D. Sterck, U. Yang, J. Heys, Reducing complexity in parallel algebraic multigrid preconditioners, SIAM J. Matrix Anal. Appl. 27 (2006) 1019–1039.
[64] E. Chow, A priori sparsity patterns for parallel sparse approximate inverse preconditioners, SIAM J. Sci. Comput. 21 (2000) 1804–1822.