Journal of Computational Physics 340 (2017) 498–518

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A scalable nonlinear fluid–structure interaction solver based on a Schwarz preconditioner with isogeometric unstructured coarse spaces in 3D

Fande Kong [a], Xiao-Chuan Cai [b,*]

[a] Modeling and Simulation, Idaho National Laboratory, P.O. Box 1625, Idaho Falls, ID 83415-3840, United States
[b] Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309-0430, United States

ARTICLE INFO

Article history:
Received 19 July 2016
Received in revised form 14 March 2017
Accepted 22 March 2017
Available online 24 March 2017

Keywords:
Nonlinear fluid–structure interaction
Unstructured mesh
Newton–Krylov–Schwarz
Isogeometric coarse mesh
Parallel scalability

ABSTRACT

Nonlinear fluid–structure interaction (FSI) problems on unstructured meshes in 3D appear in many applications in science and engineering, such as vibration analysis of aircrafts and patient-specific diagnosis of cardiovascular diseases. In this work, we develop a highly scalable, parallel algorithmic and software framework for FSI problems consisting of a nonlinear fluid system and a nonlinear solid system, that are coupled monolithically. The FSI system is discretized by a stabilized finite element method in space and a fully implicit backward difference scheme in time. To solve the large, sparse system of nonlinear algebraic equations at each time step, we propose an inexact Newton–Krylov method together with a multilevel, smoothed Schwarz preconditioner with isogeometric coarse meshes generated by a geometry preserving coarsening algorithm. Here "geometry" includes the boundary of the computational domain and the wet interface between the fluid and the solid. We show numerically that the proposed algorithm and implementation are highly scalable in terms of the number of linear and nonlinear iterations and the total compute time on a supercomputer with more than 10,000 processor cores for several problems with hundreds of millions of unknowns.

1. Introduction

Numerical simulation of nonlinear fluid–structure interaction (FSI) problems has applications in many traditional engineering fields such as vibration analysis of aircrafts, automobiles, suspension bridges and so on. More recently, the applicability of the technique has been extended to diagnosing and treatment planning of certain medical problems such as congenital and acquired cardiovascular diseases, also to the design and optimization of medical devices [1–3]. However, solving the fluid–structure interaction problems on supercomputers with a large number of processor cores is still challenging because the coupled nonlinear FSI system consists of elliptic, parabolic and hyperbolic components all in one system, and most existing algorithms and software don't scale (strong or weak) well beyond a few hundred processor cores. In this

* Corresponding author.
E-mail addresses: fande.kong@inl.gov (F. Kong), cai@colorado.edu (X.-C. Cai).

http://dx.doi.org/10.1016/j.jcp.2017.03.043
0021-9991/

paper, we provide an implementation of a multilevel unstructured Schwarz preconditioned inexact Newton–Krylov method with isogeometric coarse meshes which improve the parallel scalability of the method to over 10^4 processor cores.

The research on FSI grows rapidly [4–7]. The techniques may be broadly classified into two types: the iterative coupling approach and the monolithic coupling approach. The iterative coupling approach has been successfully applied to many engineering problems; i.e., aeroelasticity [8,9], and hemodynamics [10,11]. The monolithic coupling approach is also widely used [12–17]. For example, in [16] a Newton method with exact Jacobian solves is introduced to solve FSI problems, and the Jacobian includes the shape derivative of the fluid variables with respect to the structural motion perturbations. In [14,15] a Non-Uniform Rational Basis Spline (NURBS)-based isogeometric method is investigated for the coupled FSI problems arising from hemodynamics. Beside these literatures, there are also other important works; i.e., the particle finite element method [18,19], the immersed boundary method [7,20–23], the extended finite element method [24,25], the lattice Boltzmann method [26,27], and the meshfree method [28,29]. For parallel simulations, in [30] a software framework called Uintah is developed by introducing hybrid task-based parallelism algorithms, and the approach scales well with up to 260K cores for *structured mesh* problems. But for *unstructured mesh* problems, it is more difficult to achieve good scalability with a large number of processor cores. A parallel algorithm developed in [31] scales to several hundred processor cores, and the Newton–Krylov–Schwarz method is used in [12,13] for 2D FSI problems with several hundred processor cores, and in [17] for 3D problems with up to 3,000 processor cores. The scalability limitation of these approaches is the coarse space. In this paper, a new type of preconditioning coarse spaces is developed, and with the new coarse solver the scalability of the Newton–Krylov–Schwarz method can be extended to more than 10,000 processor cores. To the best of our knowledge, our paper is the first to obtain good scalability, in both strong and weak senses, with more than 10,000 processor cores for solving the fully coupled FSI system on *unstructured 3D meshes*.

Our algorithmic and software framework consists of several key ingredients: a monolithic coupling technique for nonlinear solid and nonlinear fluid equations, a fully implicit unstructured finite element discretization, a highly scalable Newton–Krylov–Schwarz solver with a multilevel preconditioner based on isogeometric coarse meshes, a hierarchical partitioning strategy for the fine mesh, and a well-matched partitioning approach for coarse meshes. In our approach, the incompressible Navier–Stokes equations for the fluid are formulated in an Arbitrary Lagrangian–Eulerian (ALE) framework [16,32], where a linear elasticity equation is employed to describe the deformation of the fluid domain. A nonlinear elasticity equation is described in a Lagrangian framework for the solid deformation. Three coupling conditions between the fluid and the solid are implicitly enforced in the variational forms or in the function spaces. The fluid and solid meshes are conforming on the wet interface. There are other approaches for coupling the solid and the fluid equations. For example, in [20,23] the immersed boundary method is used, and in [33,34] the coupled momentum method is proposed. Compared with other approaches, the monolithic coupling offers a more robust convergence with respect to important physics parameters [14], and eliminates the so-called added-mass effect [35] which happens when the densities of the fluid and the solid are very close to each other. In this paper, to discretize the monolithically coupled FSI system, a stabilized $P_1 - P_1$ finite element method is used for the fluid, a P_1 finite element method is used for both the solid and the fluid domain motion, and a fully implicit backward difference scheme is adopted for the time integration. The resulting nonlinear system is difficult to solve because of the highly nonlinear terms from the connective terms of the incompressible Navier–Stokes equations, the nonlinear constitutive law of the solid equation, the dependency of the fluid solution on the fluid domain motion, and the implicitly enforced coupling conditions. To overcome these difficulties, we use an inexact Newton method with an analytically computed Jacobian to solve the coupled FSI algebraic system, within which a Krylov subspace method is employed for the solution of the corresponding ill-conditioned Jacobian systems. There are also other choices for solving the coupled nonlinear systems, for example, in [36,31] the dependency of the solution on the moving fluid domain and other nonlinear terms are linearized by extrapolating the solution from the previous time steps. We do not use the simplified approach because it may become unstable for certain problems, for example, numerical instabilities occur when using the iterative coupling approach [35].

The parallel scalability of the inexact Newton–Krylov method depends almost completely on the performance of the preconditioner, especially when the number of processor cores is large. In [36], a dual threshold incomplete LU factorization and an incomplete block-LU factorization are used as preconditioners for the coupled FSI system. In [31], preconditioners are constructed based on the block-structure of an approximate Jacobian system. In this work, we introduce a multilevel Schwarz preconditioner with isogeometric coarse meshes that preserve the geometry defined by the fine surface mesh of the boundary of the computational domain and the wet interface between the fluid and the solid. This similar idea has been successfully applied for elasticity problems [37] and for fluid–structure interaction problems with linear elasticity [38]. In this paper, the nonlinear constitutive law is applied, and the algorithm has to be further improved since the resulting coupled FSI system is more challenging to solve.

The rest of the paper is organized as follows. The coupled equations and an implicit finite element method are presented in Section 2, and the monolithic, implicit inexact Newton–Krylov method with a multilevel, smoothed Schwarz preconditioner is discussed in Section 3. A geometry preserving coarsening algorithm for generating isogeometric coarse meshes is proposed in Section 4 and some numerical results for simulating blood flows in compliant arteries are given in Section 5. In Section 6, we offer some concluding remarks.

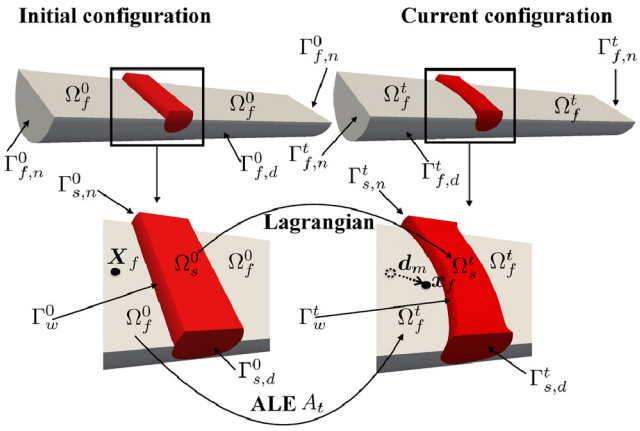

Fig. 1. An ALE mapping A_t.

2. A nonlinear coupled fluid–structure interaction problem

In this section, we describe the coupled FSI system consisting of incompressible Navier–Stokes equations for the fluid flow, a geometrically nonlinear elasticity equation for the solid, a linear elasticity equation for the fluid domain movement and three coupling conditions on the interface between the fluid and the solid. A fully implicit stabilized finite element method is used to discretize the coupled FSI system.

At time $t \in (0, T)$, let Ω_f^t be the fluid domain, Ω_s^t the solid domain, $\Gamma_{f,d}^t$ the fluid Dirichlet boundary, $\Gamma_{f,n}^t$ the fluid Neumann boundary, $\Gamma_{s,d}^t$ the solid Dirichlet boundary, $\Gamma_{s,n}^t$ the solid Neumann boundary, and Γ_w^t the wet interface. $t = 0$ corresponds to the initial configuration. We first define an ALE mapping A_t, as shown in Fig. 1, for describing the moving fluid domain:

$$A_t : \boldsymbol{x}_f = A_t(\boldsymbol{X}_f) = \boldsymbol{X}_f + \boldsymbol{d}_m, \boldsymbol{x}_f \in \Omega_f^t, \boldsymbol{X}_f \in \Omega_f^0,$$

where $\boldsymbol{d}_m$ is the fluid domain displacement at time t and it is assumed to satisfy the following linear elasticity equation

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}_m = \boldsymbol{0} & \text{in } \Omega_f^0 \times (0, T), \\ \boldsymbol{d}_m = \boldsymbol{0} & \text{on } \Gamma_f^0 \equiv \Gamma_{f,d}^0 \cup \Gamma_{f,n}^0 \times (0, T), \\ \boldsymbol{d}_m = \boldsymbol{d}_s & \text{on } \Gamma_w^0 \times (0, T), \end{cases} \tag{1}$$

where $\boldsymbol{d}_s$ is the solid displacement defined in the initial configuration and $\boldsymbol{\sigma}_m$ is the Cauchy stress tensor

$$\boldsymbol{\sigma}_m = \lambda_m \text{trace}\,(\boldsymbol{\varepsilon}_m)\, \boldsymbol{I} + 2\mu_m \boldsymbol{\varepsilon}_m, \ \boldsymbol{\varepsilon}_m = \frac{1}{2}\left(\nabla \boldsymbol{d}_m + \nabla \boldsymbol{d}_m^T\right),$$

where $\boldsymbol{\varepsilon}_m$ is the infinitesimal strain tensor and λ_m and μ_m are Lamé constants that don't have any physical meaning, and they are often chosen the same as the parameters in the solid equation to be introduced shortly. More precisely, $\lambda_m = \lambda_s$ and $\mu_m = \mu_s$, where the parameters λ_s and μ_s are defined in (3). A similar model for the domain movement is used in [39,40], which shows that the linear elasticity based model works better than the model based on the Poisson equation when the displacement is large. The ALE-based approach together with a mesh updating technique works well for problems with large deformations [41] as long as the mesh is not highly distorted. The mesh of the fluid domain may be distorted and twisted for problems involving large rotations and large rigid body motions [42]. In this situation, an effective remeshing technique is necessary to strength the ALE-based method. But, parallel remeshing is quite challenging when the number of processor cores is large. To avoid frequent remeshing, a mesh-updating technique based on the element size was considered in [40]. However, in this work, we do not consider such techniques because the deformation of the fluid domain for the blood flow does not lead to highly distorted meshes.

For the fluid, let $\boldsymbol{u}_f$ and p_f denote the velocity and pressure, respectively, and the incompressible Navier–Stokes equations on a moving domain are written as follows

$$\begin{cases} \rho_f \dfrac{\partial \boldsymbol{u}_f}{\partial t}\bigg|_{\boldsymbol{x}_f} + \rho_f \left[\left(\boldsymbol{u}_f - \dfrac{\partial \boldsymbol{d}_m}{\partial t}\right)\cdot\nabla\right]\boldsymbol{u}_f - \nabla\cdot\boldsymbol{\sigma}_f = \rho_f \boldsymbol{f}_f & \text{in } \Omega_f^t \times (0,T), \\ \nabla\cdot\boldsymbol{u}_f = 0 & \text{in } \Omega_f^t \times (0,T), \\ \boldsymbol{u}_f = \boldsymbol{v}_f^d & \text{on } \Gamma_{f,d}^t \times (0,T), \\ \boldsymbol{\sigma}_f \boldsymbol{n}_f = \boldsymbol{g}_f & \text{on } \Gamma_{f,n}^t \times (0,T), \\ \boldsymbol{u}_f = \dfrac{\partial \boldsymbol{d}_s(A_t^{-1})}{\partial t} & \text{on } \Gamma_w^t \times (0,T), \end{cases} \tag{2}$$

where $\boldsymbol{g}_f$ is a traction applied to part of boundaries, usually on the outlets, $\boldsymbol{v}_f^d$ is a velocity profile often at the inlet, $\boldsymbol{f}_f$ is a volumetric force per unit of mass, ρ_f is the fluid density, $\boldsymbol{n}_f$ is the unit outward normal vector for the fluid domain and $\boldsymbol{\sigma}_f$ is the Cauchy stress tensor for the fluid,

$$\boldsymbol{\sigma}_f = -p_f \boldsymbol{I} + 2\nu_f \boldsymbol{\varepsilon}_f,\ \boldsymbol{\varepsilon}_f = 1/2\left(\nabla \boldsymbol{u}_f + \nabla \boldsymbol{u}_f^T\right),$$

where $\boldsymbol{\varepsilon}_f$ is the strain rate tensor and ν_f is the viscosity coefficient.

We assume that the displacement $\boldsymbol{d}_s$ of the solid is governed by an unsteady, geometrically nonlinear elasticity equation [43] as follows:

$$\begin{cases} \rho_s \dfrac{\partial^2 \boldsymbol{d}_s}{\partial t^2} + \eta_s \dfrac{\partial \boldsymbol{d}_s}{\partial t} - \nabla\cdot\boldsymbol{\Pi}_s = \rho_s \boldsymbol{f}_s & \text{in } \Omega_s^0 \times (0,T), \\ \boldsymbol{\Pi}_s \boldsymbol{n}_s = \boldsymbol{g}_s & \text{on } \Gamma_{s,n}^0 \times (0,T), \\ \boldsymbol{d}_s = \boldsymbol{0} & \text{on } \Gamma_{s,d}^0 \times (0,T), \\ \boldsymbol{\sigma}_s \hat{\boldsymbol{n}}_s = -\boldsymbol{\sigma}_f \boldsymbol{n}_f & \text{on } \Gamma_w^t \times (0,T), \end{cases} \tag{3}$$

where ρ_s is the density, $\boldsymbol{g}_s$ is a traction applied to part of the boundary, $\boldsymbol{f}_s$ is a volumetric force per unit of mass, $\boldsymbol{n}_s$ and $\hat{\boldsymbol{n}}_s$ are the unit outward normal vectors under the initial configuration and the deformed domain, respectively, $\eta_s \partial \boldsymbol{d}_s/\partial t$ is the damping term used to mimic the impact of surrounding tissues, η_s is the damping parameter, and $\boldsymbol{\Pi}_s$ is the nonsymmetric first Piola–Kirchhoff stress tensor for the Saint Venant–Kirchhoff material,

$$\begin{aligned} &\boldsymbol{F} = (\boldsymbol{I} + \nabla \boldsymbol{d}_s), \\ &\boldsymbol{E} = 1/2(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I}), \\ &\boldsymbol{S} = \lambda_s \mathrm{trace}(\boldsymbol{E})\boldsymbol{I} + 2\mu_s \boldsymbol{E}, \\ &\boldsymbol{\Pi}_s = \boldsymbol{F}\boldsymbol{S}, \end{aligned}$$

where $\boldsymbol{I}$ is a 3×3 identity matrix, $\boldsymbol{F}$ is the deformation gradient tensor, $\boldsymbol{E}$ is the Green–Lagrangian strain tensor, $\boldsymbol{S}$ is the second Piola–Kirchhoff stress tensor, $\boldsymbol{\sigma}_s = \boldsymbol{\Pi}_s \boldsymbol{F}^T/det(\boldsymbol{F})$ is the Cauchy stress tensor for the solid, and μ_s and λ_s are the material Lamé constants expressed as functions of Young's modulus, E_s, and Poisson's ratio, ν_s, by

$$\mu_s = \frac{E_s}{2(1+\nu_s)} \text{ and } \lambda_s = \frac{E_s \nu_s}{(1+\nu_s)(1-2\nu_s)}.$$

Remark 1. Note that $\boldsymbol{n}_s$ and $\hat{\boldsymbol{n}}_s$ are related by the Nanson formula [43].

Remark 2. Following [17] and [44], a mass-proportional damping coefficient η_s is considered in this work to mimic the effect of the surrounding tissues. Another possible approach is to apply a special boundary condition [45].

On the wet interface, three coupling conditions are imposed to couple the solid and fluid equations. The first condition is the continuity of the velocity: $\boldsymbol{u}_f = \partial \boldsymbol{d}_s/\partial t$. The second condition is the continuity of the displacement: $\boldsymbol{d}_m = \boldsymbol{d}_s$. Lastly, the traction forces from the fluid and the solid are the same: $\boldsymbol{\sigma}_s \hat{\boldsymbol{n}}_s = -\boldsymbol{\sigma}_f \boldsymbol{n}_f$. In all, the monolithically coupled FSI equations are written as follows:

$$
\begin{cases}
\rho_f \left.\dfrac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\boldsymbol{x}_f} - \nabla\cdot\boldsymbol{\sigma}_f + \rho_f\left[\left(\boldsymbol{u}_f - \dfrac{\partial \boldsymbol{d}_m}{\partial t}\right)\cdot\nabla\right]\boldsymbol{u}_f = \rho_f \boldsymbol{f}_f & \text{in } \Omega_f^t\times(0,T),\\
\nabla\cdot\boldsymbol{u}_f = 0 & \text{in } \Omega_f^t\times(0,T),\\
\boldsymbol{u}_f = \boldsymbol{v}_f^d & \text{on } \Gamma_{f,d}^t\times(0,T),\\
\boldsymbol{\sigma}_f\boldsymbol{n}_f = \boldsymbol{g}_f & \text{on } \Gamma_{f,n}^t\times(0,T),\\
-\nabla\cdot\boldsymbol{\sigma}_m = \boldsymbol{0} & \text{in } \Omega_f^0\times(0,T),\\
\boldsymbol{d}_m = \boldsymbol{0} & \text{on } \Gamma_f^0\times(0,T),\\
\rho_s\dfrac{\partial^2\boldsymbol{d}_s}{\partial t^2} + \eta_s\dfrac{\partial \boldsymbol{d}_s}{\partial t} - \nabla\cdot\boldsymbol{\Pi}_s = \rho_s\boldsymbol{f}_s & \text{in } \Omega_s^0\times(0,T),\\
\boldsymbol{d}_s = \boldsymbol{0} & \text{on } \Gamma_{s,d}^0\times(0,T),\\
\boldsymbol{\Pi}_s\boldsymbol{n}_s = \boldsymbol{g}_s & \text{on } \Gamma_{s,n}^0\times(0,T),\\
\boldsymbol{u}_f = \dfrac{\partial \boldsymbol{d}_s(A_t^{-1})}{\partial t} & \text{on } \Gamma_w^t\times(0,T),\\
\boldsymbol{\sigma}_s\hat{\boldsymbol{n}}_s = -\boldsymbol{\sigma}_f\boldsymbol{n}_f & \text{on } \Gamma_w^t\times(0,T),\\
\boldsymbol{d}_m = \boldsymbol{d}_s & \text{on } \Gamma_w^0\times(0,T).
\end{cases}
\tag{4}
$$

To discretize (4), we consider a $P_1 - P_1$ stabilized finite element method [46,47] for the incompressible Navier–Stokes equations and a P_1 finite element method for both the solid equation and the fluid domain moving equation. Three coupling conditions are implicitly enforced in the variational forms or in the function spaces. Interested readers are referred to [17] for mote details. After the discretization in space, the corresponding semi-discretized system is a time-dependent nonlinear system as follows

$$
\frac{\partial y(t)}{\partial t} + N(y(t)) = F, \tag{5}
$$

where F is the right-hand side and $N(\cdot)$ is a nonlinear function, $y(\cdot)$ is the time-dependent vector of nodal values of the fluid velocity and pressure, and the solid velocity and displacement. Using an implicit first-order backward Euler scheme, (5) is further discretized in time as:

$$
M_n y_n + \delta t N(y_n) = \delta t F + M_n y_{n-1}, \tag{6}
$$

where δt is the time step size, y_n is the approximation at nth time step, M_n is the mass matrix dependent of y_n since the computational fluid domain is moving. With a second-order backward difference formula, (5) can also be discretized as follows:

$$
1.5M_n y_n + \delta t N(y_n) = \delta t F + 2M_n y_{n-1} - 0.5M_n y_{n-2}. \tag{7}
$$

The ALE velocity $\partial \boldsymbol{d}_m/\partial t$ is approximated by the first-order backward Euler scheme in (6) and (7). For convenience, we rewrite (6) or (7) at the nth time step as a nonlinear algebraic system:

$$
\mathcal{F}(y) = 0, \tag{8}
$$

where $\mathcal{F}(\cdot)$ is the combination of four terms in (6) or five terms in (7), and y (we drop the subscript n here for simplicity) is the vector of nodal values at the nth time step.

Equation (8) is quite difficult to solve because it has high nonlinearities from the convective term of the fluid equations, the constitutive law of the solid equation, the dependency of fluid flows on the domain movement, the three coupling conditions on the moving wet interface, more significantly, the three equations have very different mathematical characteristics; i.e., the fluid system is parabolic, the solid system is hyperbolic, and the moving domain equation is elliptic. To overcome these difficulties, an inexact Newton–Krylov method preconditioned by an unstructured multilevel Schwarz method with isogeometric coarse spaces will be introduced in the next section.

3. A fully implicit Newton–Krylov–Schwarz method

To solve the nonlinear system (8), we introduce an inexact Newton [48–50] combined with multilevel Schwarz preconditioned Krylov subspace methods. More precisely, with an initial guess $y^{(0)}$, each successive Newton step is carried out in two substeps:

$$
\begin{cases}
\mathcal{J}(y^{(k)})\delta y^{(k)} = -\mathcal{F}(y^{(k)}),\\
y^{(k+1)} = y^{(k)} + \alpha^{(k)}\delta y^{(k)}, k = 0, 1, 2, \ldots,
\end{cases}
\tag{9}
$$

Algorithm 3.1 Well-matched coarse mesh partition.

1: Input a coarse mesh Ω_{h_i}
2: Set each member in $\{\Omega_{h_i,j}\}$ as a empty set, $j = 0, 1, \cdots, np$
3: **for** $K_c \in \Omega_{h_i}$ **do**
4: Compute the center point of K_c
5: Find a fine mesh element K_f in $\Omega_{h_{L-1}}$ that contains the center point
6: Retrieve the fine mesh partition (see $\Omega_{h_{L-1},j}$) that K_f belongs to.
7: Put K_c into $\Omega_{h_i,j}$
8: **end for**
9: Output partition $\{\Omega_{h_i,j}\}$

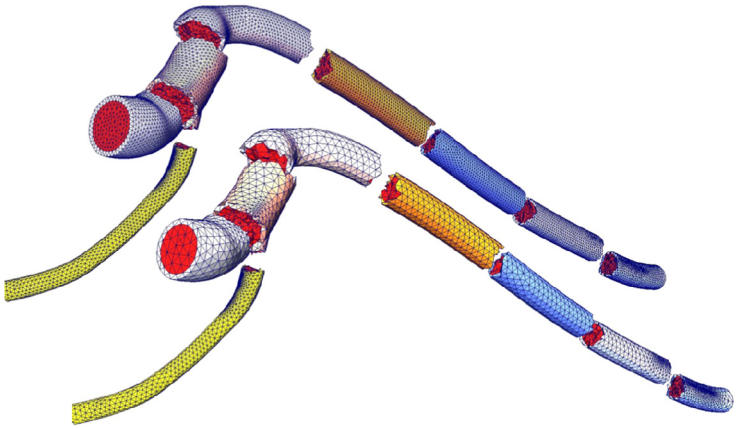

Fig. 2. A sample well-matched partition into 8 subdomains. The top figure is the fine mesh and the bottom is the coarse mesh. The subdomain may contain both fluid and solid elements. The number of degrees of freedom per subdomain is roughly the same for load balancing. The fine and coarse meshes have different triangulations on the surfaces and the wet interface just for the purpose of visualization. These meshes are not used in the simulation in Section 5.

where $\mathcal{J}(y^{(k)})$ is the Jacobian matrix evaluated at Newton step k, $\mathcal{F}(y^{(k)})$ is the residual of the nonlinear function at $y^{(k)}$, $\alpha^{(k)}$ is the step length calculated by a cubic linesearch method [51], and $\delta y^{(k)}$ is a Newton correction obtained by solving the linear Jacobian system in the first substep. An analytically computed, complete Jacobian, involving all terms of (8), often leads to a more scalable and more robust convergence of Newton than approximately constructed Jacobian using some of the terms of (8) or using a finite difference approximation. Therefore, in this work, we use the analytic form of $\mathcal{J}(y^{(k)})$ for all the calculations even though the derivation and implementation of the analytic form is time-consuming because the level of composition of some of the functions is high (more than 10) due to the domain movement and the stabilization parameters in the finite element formulation of the fluid equations. The structure of the Jacobian matrix is similar to the Jacobian matrices in our previous work [12,13]. The Jacobian problem of the fully coupled system is very ill-conditioned, and is solved by a Krylov subspace method; e.g., fGMRES [52], together with an overlapping Schwarz preconditioner.

Multilevel Schwarz is a well-understood method for elliptic problems, but for non-elliptic problems, such as (9), standard coarse spaces don't offer the best choices in the sense that they are not scalable either in terms of the number of iterations or in the total compute time. Below we first describe the method itself and then focus on a non-standard, geometry preserving, coarse space designed particularly for the FSI problems. We assume there are L triangulations of $\Omega \equiv \Omega_f^0 \cup \Omega_s^0$ denoted from coarse to fine as Ω_{h_i}, and their corresponding Jacobian matrices and vectors as $\mathcal{J}_{h_i}$, y_{h_i}, r_{h_i}, z_{h_i}, $i = 0, 1, ..., L-1$. We first partition the fine mesh $\Omega_{h_{L-1}}$ into np (np is the number of processor cores) subdomains $\Omega_{h_{L-1},j}$, then the partitions $\Omega_{h_i,j}$ of the remaining $L-1$ coarse meshes are obtained according to the fine mesh partition, that is, a coarse mesh element is allocated to the processor core that has a fine mesh element containing the center point of the coarse mesh element (see Algorithm 3.1 for details). We refer to this as a well-matched coarse mesh partition. For example, a well-matched partition with 8 subdomains is shown in Fig. 2. Note that all meshes are monolithically partitioned so that a processor may have both the solid and the fluid elements. The well-matched partitioning can greatly reduce the cross-processor data movement when performing the interpolations and restrictions between different level of meshes.

To define a one-level Schwarz preconditioner on Ω_{h_i}, let us denote the vector and matrix in each subdomain $\Omega_{h_i,j}$ as $y_{h_i,j}$ and $\mathcal{J}_{h_i,j}$, and then extend each $\Omega_{h_i,j}$ to overlap with its neighboring subdomains by a user-specified amount δ, and denote the overlapping subdomain as $\Omega^{\delta}_{h_i,j}$, the overlapping vector as $y^{\delta}_{h_i,j}$ and the restriction as $R^{\delta}_{h_i,j}$ which extracts the local overlapping vector $y^{\delta}_{h_i,j}$ from the global vector y_{h_i}, that is,

$$y^{\delta}_{h_i,j} = R^{\delta}_{h_i,j} y_{h_i} = (I \;\; 0) \begin{pmatrix} y^{\delta}_{h_i,j} \\ y_{h_i} \backslash y^{\delta}_{h_i,j} \end{pmatrix},$$

where I is an identity matrix whose size is the same as $y^{\delta}_{h_i,j}$.

Algorithm 3.2 Multilevel Smoothed Schwarz preconditioner.

1: Input a residual $r_{h_{L-1}}$ from the outer solver
2: **for** $i = L-1$ to 0 **do**
3: Let $z_{h_i}^{(0)} = 0$
4: **for** $n = 0$ to itr_i **do**
5: $z_{h_i}^{(n+1)} = z_{h_i}^{(n)} + B_{h_i}^{-1}\left(r_{h_i} - A_{h_i} z_{h_i}^{(n)}\right)$
6: **end for**
7: Set $z_{h_i} = z_{h_i}^{(itr_i)}$
8: **if** $i > 0$ **then**
9: $r_{h_{i-1}} = (I_{h_{i-1}}^{h_i})^T (r_{h_i} - A_{h_i} z_{h_i})$
10: **end if**
11: **end for**
12: **for** $i = 1$ to $L-1$ **do**
13: Let $z_{h_i}^{(0)} = z_{h_i} + I_{h_{i-1}}^{h_i} z_{h_{i-1}}$
14: **for** $n = 0$ to itr_i **do**
15: $z_{h_i}^{(n+1)} = z_{h_i}^{(n)} + B_{h_i}^{-1}\left(r_{h_i} - A_{h_i} z_{h_i}^{(n)}\right)$
16: **end for**
17: Set $z_{h_i} = z_{h_i}^{(itr_i)}$
18: **end for**
19: Output a correction $z_{h_{L-1}}$ to the outer solver

We denote $R_{h_i,j}^0$ as a non-overlapping restriction matrix that returns $y_{h_i,j}$. A one-level restricted additive Schwarz preconditioner on Ω_{h_i} [53] is defined as

$$\begin{cases} B_{h_i}^{-1} = \sum_{j=1}^{np} \left(R_{h_i,j}^0\right)^T \left(\mathcal{J}_{h_i,j}^{\delta}\right)^{-1} R_{h_i,j}^{\delta}, \\ \mathcal{J}_{h_i,j}^{\delta} = R_{h_i,j}^{\delta} \mathcal{J}_{h_i} \left(R_{h_i,j}^{\delta}\right)^T, i = 0, 1, ..., L-1. \end{cases} \tag{10}$$

To describe a multilevel preconditioner, we need an interpolation operator from Ω_{h_i} to $\Omega_{h_{i+1}}$ denoted as $I_{h_i}^{h_{i+1}}$ and its corresponding restriction operator as $\left(I_{h_i}^{h_{i+1}}\right)^T$. A multilevel Schwarz preconditioner [54,55] is described in Algorithm 3.2, where itr_i is the number of iterations for Ω_{h_i}. In practice, itr_i is usually controlled by a tolerance or by a pre-determined maximum number of iterations. To further improve the algorithm, Richardson procedures at line 5 and 15 can be replaced by other iterative methods, such as GMRES [56].

The performance of Algorithm 3.2 depends critically on how the coarse meshes are constructed and how the coarse linear systems are solved. If the fine mesh is structured, the coarse meshes can be generated easily even when the number of processor cores is large [57]. But if unstructured meshes have to be used for a complex geometry, the implementation of the multilevel method is tricky, especially when number of processor cores is large because both partitioning and coarsening of the fine mesh are highly nontrivial. To partition the fine mesh, we employ a hierarchical partitioning strategy since most existing partitioners; e.g. ParMETIS/METIS [58], do not work well when the number of processor cores is large and the mesh is highly irregular (i.e., the degree of the connectivity graph changes drastically from point to point). The strategy consists of two steps: (1) use ParMETIS/METIS to partition the fine mesh into np_1 (np_1 is often the number of compute nodes) subdomains; (2) further partition each subdomain by METIS into np_2 (np_2 is often the number of cores on a compute node) smaller subdomains. We finally have $np = np_1 \times np_2$ subdomains in total. To coarsen the fine mesh, we propose a geometry preserving coarsening algorithm, in the next section, that produces several isogeometric coarse meshes which keep the geometry of the fine mesh on the boundaries and the wet interface.

4. Isogeometric coarse spaces

It is straightforward to apply Algorithm 3.2 to problems on *structured meshes* since coarse meshes can be obtained using a uniform refinement or coarsening without losing the geometry of the computational domain. But it becomes tricky when solving problems defined on *unstructured meshes* because the construction of scalable coarse meshes is highly nontrivial. For this sake, algebraic multigrid (AMG) is widely used [59], and it has been implemented in several popular libraries [60,61]. But there are definite advantages in a geometric multilevel method [62]. We hence construct the multilevel Schwarz method geometrically by introducing a new mesh coarsening algorithm which produces coarse meshes with preserving the geometry of the computational domain and the wet interface. We refer to those coarse meshes as “isogeometric coarse meshes” and the new coarsening algorithm as “geometry preserving coarsening algorithm”. The isogeometric coarse meshes share exactly the same geometry on the boundary and the wet interface as the fine mesh, and the geometric features on coarse meshes are crucial for making the overall algorithm scalable. In our experience, the algorithm is not scalable and often does not converge for 3D FSI problems if the geometric features are not preserved on the coarse meshes. However, isogeometric coarse meshes are not easy to produce because preserving the geometry from the fine mesh is a harsh requirement for most

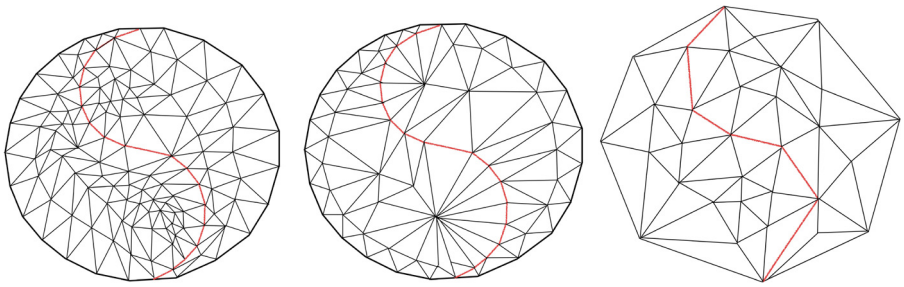

Fig. 3. Sample coarse meshes. Left: fine mesh; middle: isogeometric coarse mesh; right: coarse mesh without preserving geometry. Wet interface is red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 4.1 Vertex selecting procedure.

1: $\mathcal{S}^{(0)} = \mathcal{S}_C$
2: $\mathcal{S}^{(1)} = \mathcal{S}^{(0)} + \mathcal{S}_F$
3: $\mathcal{S}^{(2)} = \mathcal{S}^{(1)} + \mathcal{S}_W$
4: Apply the MIS of conflict graph to $\mathcal{S}_I$ to get $\tilde{\mathcal{S}}_I$
5: $\mathcal{S} = \mathcal{S}^{(2)} + \tilde{\mathcal{S}}_I$
6: Output $\mathcal{S}$

existing coarsening algorithms. To overcome this difficulty, we below describe the geometry preserving coarsening algorithm using which several isogeometric coarse meshes are produced. An example is shown in Fig. 3, where the isogeometric coarse mesh shares exactly the same geometry as the fine mesh while its interior is much coarser than the fine mesh for saving the computational cost on the coarse level, and the coarse mesh obtained by uniformly coarsening the fine mesh does not preserve the geometry.

The geometry preserving coarsening algorithm consists of two steps: (1) select a subset of vertices $\mathcal{S}$ to keep and (2) delete unwanted vertices $\mathcal{D} = \mathcal{M}/\mathcal{S}$ ($\mathcal{M}$ represents all the vertices in the fine mesh) from the fine mesh. To preserve the exact geometry of the fine mesh, all vertices on the curved boundary and the wet interface are kept, and to make the coarse level problem cheap to compute, the vertices in the interior of the fine mesh are coarsened based on the maximum independent set algorithm (MIS) [62–65]. We also keep all the vertices on the flat boundary for simplifying the implementation of the algorithm. The mesh quality may be degraded soon when using the MIS of the vertex connectivity graph. To fix this issue, a better graph constructing strategy is introduced and discussed in [64], which takes not only the topology but also the geometry of the mesh into account. The graph constructed based on this idea is called "conflict graph". We coarsen the interior of the fine mesh based on the conflict graph, but follow a simpler graph constructing idea in [65]. The interested reader is referred to [37,64,65] for more details. For convenience, we denote the vertices on the curved boundary as $\mathcal{S}_C$, those on the flat boundary as $\mathcal{S}_F$, the wet interface vertices as $\mathcal{S}_W$ and the interior vertices as $\mathcal{S}_I$. A vertex selecting algorithm is described in Algorithm 4.1. Note that Algorithm 4.1 is different from those in [62,65]. In Algorithm 4.1, all vertices on the boundaries and the wet interface are kept to preserve the geometry from the fine mesh exactly, while in [62,65], the general shape of the computational domain is considered only. Note that Algorithm 4.1 is different from that in [37] which doesn't consider any internal boundary.

Based on $\mathcal{S}$, there are two approaches to construct a coarse mesh. The first one is to generate a coarse mesh, from the scratch, that includes the vertices in $\mathcal{S}$ only and approximates the domain spanned by the fine mesh. This remeshing procedure can be accomplished using standard meshing techniques, e.g. Delaunay triangulation [66], Constrained Delaunay triangulation [67]. The second approach is to incrementally remove unwanted vertices in $\mathcal{D}$ one by one [62,65]. We use the second approach because it is more robust than most existing algorithms in the sense that the second approach guarantees that at least one valid coarse mesh is produced. We refer to the second approach as "incremental deleting algorithm". More precisely, the incremental deleting algorithm takes several sweeps of $\mathcal{D}$, and in each sweep, we delete every removable vertex in $\mathcal{D}$ using an edge-contraction operation [65]. After each sweep, a mesh-smoothing approach [68,69] is applied to the current coarse mesh, some unremovable vertices in the current sweep may become removable after the application of the mesh-smoothing, and they will be revisited again in the next sweep. This procedure repeats until all or most vertices in $\mathcal{D}$ have been deleted. From our experiences, the number of left vertices is a small proportion of $\mathcal{D}$ even though for problems with complex geometry.

In the incremental vertex deletion algorithm, the edge-contraction approach is utilized to remove unwanted vertices from the fine mesh, and the basic idea is to delete a vertex by shrinking one of its incident edges to zero length as shown in Fig. 4, where the left picture is the fine mesh and the right picture is the coarse mesh produced by removing vertex v_1 from the fine mesh. To delete v_1, we slide it along the edge e_{15} to v_5. In this procedure, edges e_{61}, e_{41} are first merged to edges e_{65}, e_{45}, then edges e_{21}, e_{31} are extended as edges e_{25}, e_{35}, and then five incident elements K_{134}, K_{145}, K_{156}, K_{162}, K_{123} are removed and lastly three new elements K_{562}, K_{523}, K_{534} are formed.

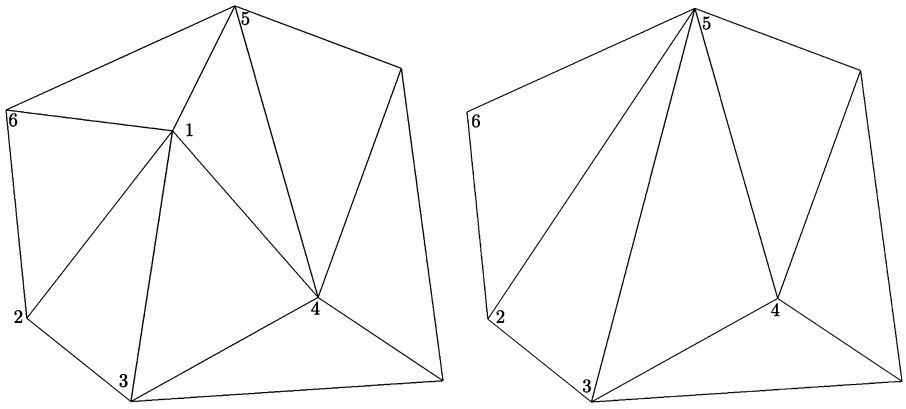

Fig. 4. Deleting a vertex based on an edge-contraction approach.

To remove a vertex, there exists several possible edge-contractions, but not all of them are valid. For example, if we slide v_5 along e_{54} to v_4 to delete v_5, an invalid element K_{614} with a negative area (which means the element is inverted) is constructed. We hence can not delete v_5 from the current mesh by this edge-contraction, but it may be removed by another edge-contraction, for example, sliding along e_{51}. In this paper, the edge selection for the edge-contraction is based on the following criteria:

1. An edge-contraction is chosen only if no inverted elements are formed in the resulting coarse mesh.
2. We can slide a vertex along any edge among its m incident edges. For example in Fig. 4, we can slide v_1 along e_{21}, e_{31}, e_{41}, e_{51} or e_{61}. We choose the edge-contraction which leads to the highest-quality coarse submesh among the submeshes spanned by m incident vertices.

For a vertex, if we can not find an edge-contraction which satisfies the Condition 1, we assume that the vertex can not be removed in the current sweep. But it might be deleted in the next sweep after applying a mesh-smoothing algorithm to improve the current mesh.

There are two approaches to improve the mesh. The first approach is to improve the connectivity of vertices by swapping faces and edges. The second approach is to adjust the locations of vertices. In this paper, we use the second approach, which is often referred to as mesh-smoothing, to change locations of interior vertices while fixing vertices on the boundary and the wet interface to preserve the geometry. The combination of a Laplacian smoothing [68] and an optimization-based smoothing [69,70] is used in this paper. The basic idea of the Laplacian smoothing is to adjust the location of a vertex v to the arithmetic mean of its neighboring vertices $\mathcal{N}_v$. More precisely, the new location ξ^* of v is obtained as follows:

$$\xi^* = \frac{1}{s_v}\sum_{i=1}^{s_v}\xi_i,$$

where ξ_i is the coordinate of $v_i \in \mathcal{N}_v$ and s_v is the size of $\mathcal{N}_v$. In practice, the Laplacian smoothing is cheap, but it does not guarantee that the mesh is improved and the resulting mesh is valid. To avoid this issue, we use an optimization-based smoothing algorithm to adjust the location of v by optimizing a quality measuring function that depends on the dihedral angles, solid angles and aspect ratios. To describe the algorithm, let us denote the incident elements of v as $\mathcal{K}_v$ and the subdomain spanned by $\mathcal{K}_v$ as Ω_v, and define a set of dihedral angle functions for $\mathcal{K}_v$ as $\gamma(\xi)_i, \xi \in \Omega_v$, $i = 1, 2, \cdots, s_\gamma$. Here s_γ is the number of dihedral angles in $\mathcal{K}_v$. $s_\gamma = 6s_k$, where s_k is the number of elements in $\mathcal{K}_v$, because there are six dihedral angles in a 3D tetrahedral element. The optimization problem is defined as:

$$\max_{\xi\in\Omega_v}\min_{1\le i\le s_\gamma}\sin(\gamma(\xi)_i), \tag{11}$$

which we solve by using the method of steepest descent [71], and the optimal solution ξ^* is used as the adjusted location of v. The new mesh-smoothing often offers a better mesh, but solving (11) is expensive. In the actual computation, we hence combine the Laplacian and the optimization-based smoothing, that is, we first use the Laplacian smoothing to adjust locations, and the optimization-based smoothing is performed if the Laplacian smoothing fails to improve $\mathcal{K}_v$.

Remark 3. Mesh-smoothing is applied to interior vertices only, and no mesh-smoothing is used to the vertices on the boundaries and the wet interface to preserve the geometry.

The incremental deleting algorithm is summarized in Algorithm 4.2. Note that Algorithm 4.2 is different from that in [65]. At the line 15 of Algorithm 4.2, the mesh-smoothing approach is used only, while in [65], both the mesh-smoothing and an

Algorithm 4.2 Incremental deleting algorithm.

Initialize ns as a large number
Set $nd = 0$
Set nr as the size of $\mathcal{D}$
Set $\mathcal{D}_r = \mathcal{D}$
Set $\mathcal{D}_d = \emptyset$
for $n = 0$ to ns **do**
  for $v \in \mathcal{D}_r$ **do**
    if v is removable **then**
      Delete v using the edge-contraction algorithm
      $nd = nd + 1$
      $nr = nr - 1$
      Add v into $\mathcal{D}_d$
    end if
  end for
  Apply the mesh-smoothing algorithm to the current coarse mesh
  if $nd = 0$ or $nr = 0$ **then**
    Break
  end if
  $nd = 0$
  $\mathcal{D}_r = \mathcal{D}_r / \mathcal{D}_d$
  Set $\mathcal{D}_d = \emptyset$
end for
Output the current coarse mesh

edge/face swapping [69] are adopted for generating a high quality coarse mesh in the traditional sense. The quality of the coarse mesh produced using Algorithm 4.2 is acceptable because it offers a scalable coarse component of the preconditioner and makes the overall NKS scalable to more than 10,000 processor cores. We also need to point out that the quality of the coarse mesh does not affect the overall solution accuracy because the overall solution accuracy is determined by the fine mesh only.

5. Numerical experiments

In this section, we provide two examples to test the parallel scalability of the preconditioners with respect to different number of processor cores and the robustness of the proposed algorithms with respect to different physics parameters. The first example consists of a cylinder as the fluid domain and a small flexible cylinder inside the other cylinder as the solid domain. The second example is a patient-specific pulmonary artery provided to us by the University of Colorado Medical School. For time discretization, an implicit backward Euler scheme is used for the first example and a second-order backward difference formula is chosen for the second example. The fully implicit solver and the geometry preserving coarsening algorithm are implemented based on PETSc [60] and GRUMMP [72], respectively. The fine meshes are generated using CUBIT [73]. The computations are carried out on an IBM iDataPlex cluster consisting of 2.6-GHz Intel Xeon E5-2670 (Sandy Bridge) processors, 2 GB memory per core and FDR Mellanox InfiniBand interconnect with Bandwidth 13.6 GBps. There are 16 processor cores per compute node, and all 16 processor cores are used in the calculation.

The stopping conditions for the inexact Newton and the outer linear solver are $rtol_n = 10^{-6}$ and $rtol_l = 10^{-4}$ (based on the vector L^2 norm), respectively. We use $\delta = 1$ as the subdomain overlapping size for the Schwarz preconditioner and ILU(1) (incomplete LU factorization with one level of fill-ins) as the subdomain solver. fGMRES (outer linear solver) restarts at 100. The number of GMRES iterations on the coarsest level is controlled by the maximum number of iterations (100) or a relative tolerance 10^{-10}, and the number of GMRES iterations on the second coarse level (if we use the three-level method) is determined by the maximum number of iterations (10) or a relative tolerance 10^{-5}. We use 1 and 10 iterations of GMRES as solvers on the finest level for the first and the second examples, respectively. "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for the entire simulation, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "MEM" is the estimated memory usage per processor core in MB, "SP" is the speedup and "EFF" is the parallel efficiency.

5.1. A benchmark case

The setup of a 3D FSI benchmark problem, as shown in the top of Fig. 1, consists of a big straight cylinder as the fluid domain with length 5 cm and radius 0.5 cm, and a small flexible cylinder as the solid domain with length 1 cm and radius 0.2 cm, which is immersed in the fluid domain. A time dependent traction $1.33 \times 10^4 \times (1 - 1.5 \times 10^5 \times (t - 0.0025)^2)$ dyn/cm^2 is imposed on the inlet for 5 ms, and a traction-free boundary condition is applied to the outlet. The fluid is characterized with viscosity $\nu_f = 0.001$ g/(cm s), and density $\rho_f = 1.0$ g/cm^3. The parameters for the solid are the

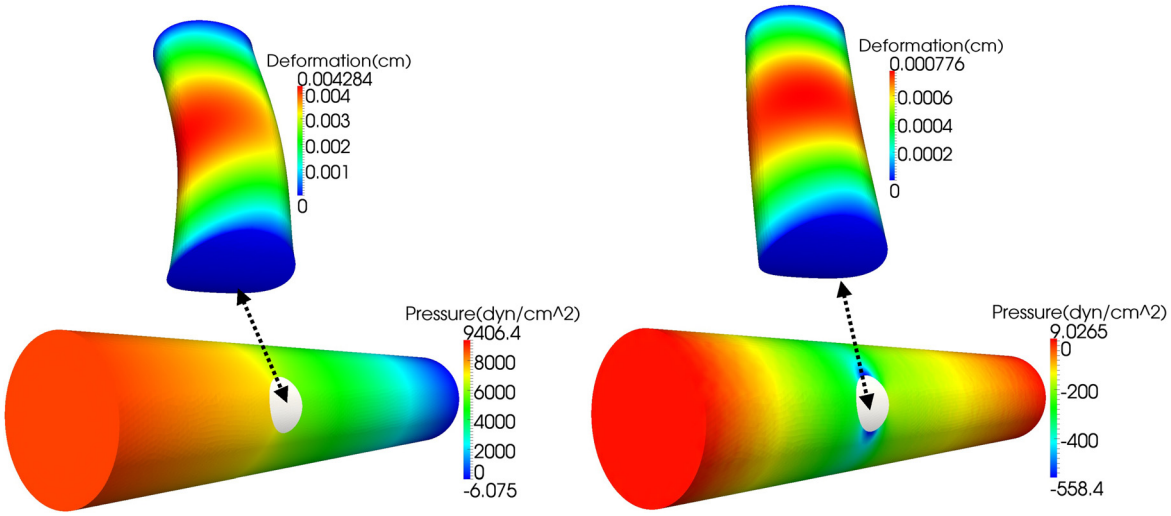

Fig. 5. Numerical solution at $t = 4\ \mathrm{ms}, 7\ \mathrm{ms}$ for the benchmark case. Top: solid deformation; bottom: fluid pressure.

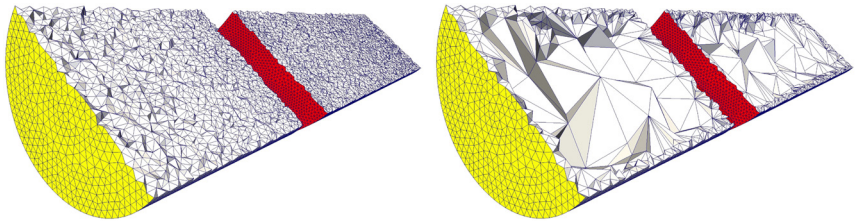

Fig. 6. Left: fine mesh; right: isogeometric coarse mesh. Note that the triangulation of the coarse mesh on the boundary (yellow) and the wet interface (red) is the same as the fine mesh, but it is much coarser in the interior. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Young's modulus $E = 3 \times 10^6\ \mathrm{g/(cm\,s^2)}$, the Poisson ratio $\nu_s = 0.3$, and the solid density $\rho_s = 1.2\ \mathrm{g/cm^3}$. Numerical solution at $t = 4\ \mathrm{ms}, 7\ \mathrm{ms}$ is shown in Fig. 5. $\delta t = 10^{-4}$ s is used as the time step size in this test, unless otherwise specified. A sample isogeometric coarse mesh is shown in Fig. 6.

(1) Impact of subdomain overlap. Let us first explore the impact of subdomain overlap sizes. Larger overlap often leads to a faster convergence of the Schwarz method, but at the same time it increases the computation and communication for each iteration. To achieve an optimal convergence in terms of the total compute time, we test the parallel performance using different overlap sizes. We use a two-level method, where the fine mesh has 1,251,813 vertices, 7,363,015 elements and 8,220,496 unknowns, and the coarse mesh has 216,380 vertices, 1,048,162 elements and 1,084,152 unknowns. The time step size is set as $\delta t = 10^{-3}$ s, and the simulation is carried out for 10 time steps. Numerical results are summarized in Table 1.

In Table 1, we see clearly from the eighth column (MEM) that more memory is used when we increase the overlap because local subproblems become larger. The number of linear iterations, LI, per Newton step decreases with the increase of the subdomain overlap, which indicates that larger overlap size often leads to better convergence. But this does not have to result in a better performance in terms of the total compute time because each iteration takes more time. The preconditioner setup time, PSetup, is often longer with larger overlap. The preconditioning time, including the application and the setup of the preconditioner, is determined by the number of linear iterations. The number of Newton iterations stays as a constant for all cases. $\delta = 1$ and $\delta = 2$ have similar performance in terms of the total compute time, and $\delta = 0$ needs more linear iterations, which results in more compute time. All cases have good parallel scalabilities in terms of the total compute time when increasing the number of processor cores from 128 to 1,024. We will use $\delta = 1$ in the following tests since it uses less memory compared with $\delta = 2$ even though they have similar performance.

(2) Different isogeometric coarse meshes. For convenience, let us denote the fine mesh as Ω_{h_3}. We coarsen Ω_{h_3} to produce a coarse mesh denoted as Ω_{h_2} which is further coarsened to generate the second coarse mesh Ω_{h_1}. Similarly, we get the third coarse mesh Ω_{h_0} by coarsening Ω_{h_1}. The fine mesh Ω_{h_3} has 1,251,813 vertices, 7,363,015 elements and 8,220,496 unknowns, the first coarse mesh Ω_{h_2} has 392,253 vertices, 2,099,948 elements and 2,296,434 unknowns, the second coarse mesh Ω_{h_1} has 216,380 vertices, 1,048,162 elements and 1,084,152 unknowns, and the third coarse mesh Ω_{h_0} has 165,974 vertices, 736,561 elements and 736,529 unknowns. We combine the fine mesh Ω_{h_3} with $\Omega_{h_i}, i = 0, 1, 2$, respectively, to form three two-level methods. We compare the performance of three two-level methods. The time step

Table 1
Impact of overlap. A coupled FSI system with 8,220,496 unknowns is solved by a two-level Schwarz preconditioned inexact Newton–Krylov method. *np* is the number of processor cores (MPI tasks), δ is the subdomain overlap size, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "MEM" is the estimated memory usage per processor core in MB, "SP" is the speedup and "EFF" is the parallel efficiency.

np	δ	NI	LI	T	PSetup	TPrec	MEM	SP	EFF
128	0	4	6.9	1301	159	748	521	1	100%
128	1	4	4.5	1196	177	647	596	1	100%
128	2	4	4	1190	189	641	687	1	100%
256	0	4	7.4	653	27	321	271	1.99	99%
256	1	4	4.4	574	40	244	307	2	100%
256	2	4	4	580	56	250	349	2	100%
512	0	4	7.5	368	17	201	144	3.5	88%
512	1	4	4.4	318	23	152	173	3.76	94%
512	2	4	4	367	91	200	209	3.24	81%
1,024	0	4	8	236	36	151	84	5.5	69%
1,024	1	4	4.4	210	45	124	109	5.71	71%
1,024	2	4	4.1	189	23	104	144	6.29	79%

Table 2
Different isogeometric coarse meshes. A coupled FSI system with 8,220,496 unknowns is solved by three two-level Schwarz preconditioned inexact Newton–Krylov methods, respectively. *np* is the number of processor cores (MPI tasks), "mesh" represents the coarse mesh of the two-level method, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "SP" is the speedup and "EFF" is the parallel efficiency.

np	mesh	NI	LI	T	PSetup	TPrec	SP	EFF
128	Ω_{h_2}	4	3.4	1239	94	584	1	100%
128	Ω_{h_1}	4	4.4	1065	83	420	1	100%
128	Ω_{h_0}	4	12	1389	83	730	1	100%
256	Ω_{h_2}	4	3.4	652	44	325	1.9	95%
256	Ω_{h_1}	4	4.4	574	40	244	1.85	92%
256	Ω_{h_0}	4	12	748	37	417	1.86	93%
512	Ω_{h_2}	4	3.5	363	28	202	3.4	85%
512	Ω_{h_1}	4	4.4	318	23	152	3.35	84%
512	Ω_{h_0}	4	12	440	24	273	3.16	79%
1,024	Ω_{h_2}	4	3.7	233	47	149	5.3	66%
1,024	Ω_{h_1}	4	4.4	210	45	124	5.1	64%
1,024	Ω_{h_0}	4	12.7	308	45	223	4.5	56%

$\delta t = 10^{-3}$ is used and the simulation is carried out for 10 time steps in this test. Numerical results are summarized in Table 2.

From Table 2, we see easily that all three isogeometric coarse meshes offer a good coarse component for the two-level preconditioner in the sense that the number of Newton iterations and the number of fGMRES iterations stay as constants when we increase the number of processor cores. The coarse mesh Ω_{h_1} has the best performance in terms of the total compute time. We observe that, for this test case, the quality of the coarse mesh does not affect the performance of the preconditioner much. For example, in Table 2, the quality of Ω_{h_1} is worse than Ω_{h_2} because Ω_{h_1} is obtained by coarsening Ω_{h_2}, but the algorithm using Ω_{h_1} has a better performance than that obtained by using Ω_{h_2}. All cases are scalable in terms of the compute time, especially when number of processor cores is smaller than 1,024. The problem is relatively small when using 1,024 processor cores so that the parallel efficiency is a little low. This issue will be discussed later in the study of parallel scalability.

(3) Robustness with respect to physical parameters. The fluid–structure interaction problem consists of the fluid, the solid and the fluid domain motion equations that have important parameters including the fluid density ρ_f, Poisson ratio ν_s, Young's modulus E_s and the solid density ρ_s, Poisson ratio ν_m and Young's modulus E_m for the fluid domain movement. In the simulations, we set $\nu_m = \nu_s$ and $E_m = E_s$. In [35], it is reported that the convergence becomes more difficult to achieve if the densities of the fluid and the solid are close, when a loosely coupled approach is used. Conversely, the ALE-based monolithic coupling algorithm has no such issues and features a good convergence under the same conditions, which is verified numerically in Table 3. We test the robustness of the proposed algorithm with respect to ν_s, E_s, ρ_s, and ρ_f. The same coarse and fine meshes used in the study of Schwarz overlap are chosen, and the experiment is run on 1,024 processor

Table 3
Robustness with respect to physical parameters. A coupled FSI system with 8,220,496 unknowns is solved by a two-level Schwarz preconditioned inexact Newton–Krylov method. ν_s is the Poisson's ratio, E_s is the Young's modulus, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec.

ν_s	E_s	NI	LI	T	PSetup	TPrec
0.3	10^6	2.8	6.3	175	38	115
0.4	10^6	2.7	7.6	184	37	127
0.45	10^6	2.6	10.8	210	36	154
0.46	10^6	2.6	11.6	217	36	161
0.47	10^6	2.5	12.6	219	34	164
0.48	10^6	2	16.8	211	28	167
0.3	10^3	2.2	3.5	117	30	69
0.3	10^4	2.3	3.4	121	32	71
0.3	10^5	2.5	13.4	230	35	175
0.3	10^6	2.8	6.3	174	38	115
0.3	10^7	2	4.3	112	28	68
0.3	10^8	2	4.5	113	28	70
ρ_f	ρ_s	NI	LI	T	PSetup	TPrec
0.1	1.2	3.4	5.8	194	38	123
0.5	1.2	3.1	4.4	160	34	95
1	1.2	2.8	4.4	146	31	86
5	1.2	2.1	4.8	114	23	68
10	1.2	2	9.3	143	22	99
1	0.1	2.7	4.7	144	30	86
1	0.5	2.8	4.3	144	31	85
1	1	2.8	4.5	146	31	87
1	5	2.8	4.8	149	31	89
1	10	2.8	4.7	148	31	89

cores. ILU(2) is used as the subdomain solver on the fine level when we test the robustness with respect to ν_s and E_s, and ILU(1) is chosen for other tests. Numerical results on the parameters ν_s, E_s, ρ_s, and ρ_f are summarized in Table 3.

In the first part of Table 3, the number of Newton iterations decreases and the number of fGMRES iterations increases when we increase ν_s from 0.3 to 0.48. The corresponding TSetup stays close to some constants because the decrease of the number of Newton iterations is small, and meanwhile the TPrec increases due to the increase of the number of linear iterations. T increases when we increase Poisson's ratio because the corresponding problems become more difficult to solve. The proposed approach works well even when ν_s is close to 0.5, and the performance in terms of the compute time is maintained when using different Poisson ratios. Both numbers of Newton iterations and fGMRES iterations increase first and then decrease a little when increasing Young's modulus from 10^3 to 10^8. The compute times are similar except $E_s = 10^5$ and $E_s = 10^6$, where more compute time is spent on solving nonlinear and linear systems. TPrec and PSetup times are similar except at $E_s = 10^5$ and $E_s = 10^6$, where more compute time is required because of the increase of the number of linear iterations. The algorithm is quite robust with respect to E_s because the coupled FSI systems corresponding to a wide range of E_s is solved well. When ρ_f increases from 0.1 to 10, fewer Newton iterations are required and the number of fGMRES iterations is a constant except when $\rho_f = 10$, where the number of fGMRES iterations is almost doubled. The compute time decreases until $\rho_f = 5$, and it is increased a little when $\rho_f = 10$. When we increase ρ_f, PSetup decreases because of the reduction of the number of Newton iterations, and TPrec decreases as well because the reduction of the number of linear iterations except at $\rho_f = 10$, where TPrec is increased to 99 seconds because the corresponding number of linear iterations is almost doubled. With using different ρ_s, both the number of Newton iterations and the number of fGMRES iterations are kept as constants so that the compute time is almost a constant. The algorithm is not sensitive to ρ_s in this test, and TPrec and PSetup stay close to some constants. Note that the algorithm works well even when the densities are the same.

In all, the proposed approach is very robust with respect to these important physical parameters in terms of the numbers of Newton iterations and fGMRES iterations, and the total compute time.

(4) Strong scalability. There are many choices of subdomain solvers for the multilevel Schwarz preconditioner. The most popular choice is an incomplete LU (ILU) factorization. We can use different levels of fill-ins which often affect the convergence of the algorithm. With different subdomain solvers, we test the parallel scalability of the propose approach with up to 10,240 processor cores. A three-level Schwarz preconditioner is used in this test. The fine mesh has 53,789,339 vertices, 320,283,584 elements and 365,890,690 unknowns, the first coarse mesh has 854,079 vertices, 5,004,431 elements and 5,575,441 unknowns, and the coarsest mesh has 160,951 vertices, 774,998 elements and 797,453 unknowns. The two coarse meshes are much coarser than the fine mesh in order to save the compute times on the coarse levels. 10 iterations of GMRES is used as a solver on the second coarse level. Poisson ratio $\nu_s = 0.4$ and Young's modulus $E_s = 10^7$ are chosen as

Table 4
Parallel scalability with up to 10,240 processor cores. A coupled FSI system with 365,890,690 unknowns is solved by a three-level Schwarz preconditioned inexact Newton–Krylov method. *np* is the number of processor cores (MPI tasks), "subsolve" represents the subdomain solve on the fine mesh, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "MEM" is the estimated memory usage per processor core in MB, "SP" is the speedup and "EFF" is the parallel efficiency.

np	subsolve	NI	LI	T	PSetup	TPrec	MEM	SP	EFF
4,096	ILU(1)	2.9	10.9	1112	46	423	905	1.00	100%
4,096	ILU(2)	2.3	14.2	1672	111	1090	1271	–	–%
6,144	ILU(1)	2.9	16.5	853	30	379	703	1.30	87%
6,144	ILU(2)	2.8	8.8	803	88	350	870	1.38	92%
8,192	ILU(1)	2.9	12.5	625	23	255	555	1.78	89%
8,192	ILU(2)	2.6	9.5	598	61	264	724	1.86	93%
10,240	ILU(1)	2.7	15.8	535	20	251	516	2.08	83%
10,240	ILU(2)	2.8	7.9	508	53	227	590	2.19	88%

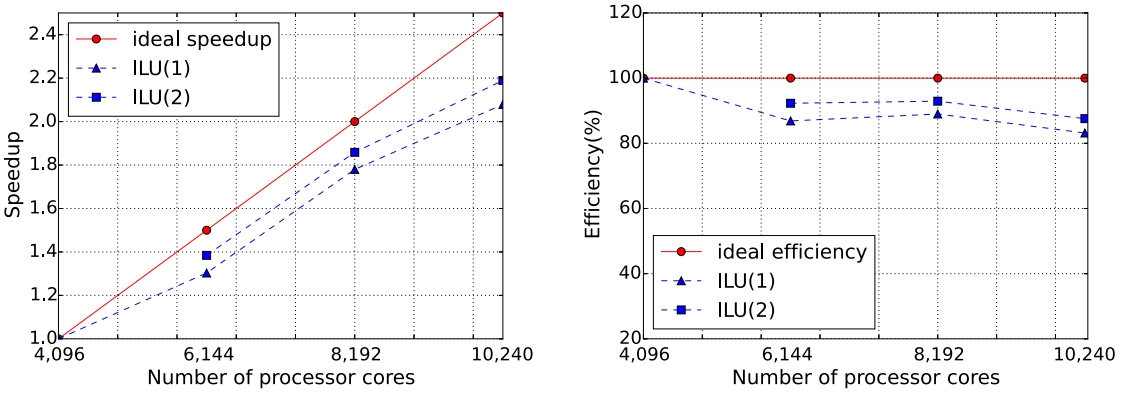

Fig. 7. Speedup and parallel efficiency using different subdomain solvers for the benchmark case.

solid parameters in this test. The parallel scalability is shown in Table 4, and the corresponding speedups and the parallel efficiencies are plotted in Fig. 7.

In Table 4, the speedups and the parallel efficiencies are calculated using the compute time of the best subdomain solver as a basis when we use 4,096 processor cores, because it is straightforward to show the performance difference of different subdomain solvers. ILU(1) is better when we are using 4,096 processor cores since ILU(2) require too much memory, which makes the algorithm slow in terms of the compute time. Continuing increasing the number of processor cores can resolve this memory issue. Hence, the performance of ILU(2) in terms of the compute time is better when we use 6,144, 8,192 and 10,240 processor cores. For ILU(1) and ILU(2), the number of Newton iterations and the number of fGMRES iterations remain as constants when increasing the number of processor cores from 4,096, 6,144, 8,192, to 10,240, which indicates that the approach is mathematically scalable. PSetup and TPrec are also scalable in the sense that they are proportionally reduced when we increase the number of processor cores. An important conclusion is that in order to make the overall algorithms scalable, all critical components of the algorithm have to be scalable. In summary, both subdomain solvers lead to a scalable approach in terms of the compute time and the numbers of Newton iterations and fGMRES iterations. The corresponding speedups and parallel efficiencies are also plotted in Fig. 7. To further understand the algorithm, the compute times spent on different levels of the three-level preconditioner, interpolation/restriction and outer solvers are drawn in Fig. 8. "level 0" is the coarsest level, "level 1" the second coarse level and "level 2" is the fine level. Compute times on the outer solvers and the fine level are scalable, and the second coarse and the coarsest levels are not scalable when we use 10,240 processor cores. Fortunately, the compute times on the coarsest and the second coarse levels take only a small portion of the total compute time. This phenomena implies that more levels may be required if we continue increasing the number of processor cores from 10,240. The compute time on the interpolation/restriction accounts only a small portion of the total compute time, which indicates that the well-matched partitioning approach is effective on minimizing the data movement between different levels.

(5) Weak scalability. We test the weak scaling of the three-level Schwarz preconditioner. In the tests, the number of elements of the mesh is increased proportionally to the number of processor cores. Note that, for *unstructured meshes*, it is not always possible to maintain the exact proportionality, for example, the number of elements of the larger mesh is approximately twice the number of elements of the old mesh when we double the number of processor cores. For this test,

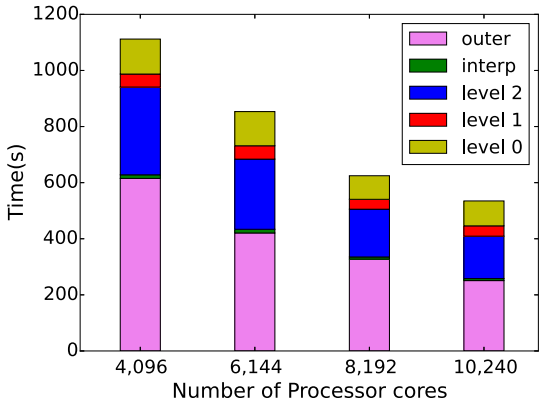

Fig. 8. Compute times on different levels of the multilevel preconditioner and outer solvers for the benchmark case when we use ILU(1). "level 0" is the coarsest level, "level 1" is the second coarse level and "level 2" is the fine level. "outer" represents the outer solver including the nonlinear and the linear solver. "interp" denotes the interpolation/restriction.

Table 5
Mesh details for different processor counts. ne_{h_i}, nn_{h_i} and nu_{h_i}, $i = 0, 1, 2$ are the numbers of elements, nodes and unknowns, respectively. np is the number of processor cores.

$np =$	2,048	4,096	8,192	12,288
ne_{h_2}	21,927,384	51,664,992	92,829,280	131,295,840
nn_{h_2}	3,716,553	8,722,691	15,631,513	22,071,078
nu_{h_2}	24,857,471	58,634,167	105,370,077	151,002,482
ne_{h_1}	2,740,923	6,458,124	11,603,660	16,411,980
nn_{h_1}	470,272	1,099,509	1,965,430	2,768,811
nu_{h_1}	3,046,055	7,204,971	12,972,702	18,599,152
ne_{h_0}	392,208	948,989	1,464,760	1,941,788
nn_{h_0}	86,040	196,228	300,497	394,122
nu_{h_0}	393,859	979,754	1,515,808	2,054,874

Table 6
Weak scaling with up to 12,288 processor cores. A three-level Schwarz preconditioned inexact Newton–Krylov method is used. np is the number of processor cores (MPI tasks), "RoE" denotes the ratio of the number of elements of the large mesh to the base mesh, "subsolve" represents the subdomain solve on the fine mesh, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "MEM" is the estimated memory usage per processor core in MB, "SP" is the speedup and "EFF" is the parallel efficiency.

np	RoE	subsolve	NI	LI	T	PSetup	TPrec	MEM	EFF
2,048	1	ILU(1)	2.8	5.5	165	10	86	175	100%
2,048	1	ILU(2)	2.8	5.8	191	20	111	217	–%
4,096	2.36	ILU(1)	2.5	5.2	188	20	102	201	104%
4,096	2.36	ILU(2)	2.4	5.5	203	33	120	243	96%
8,192	4.23	ILU(1)	2.5	4.7	208	26	125	213	84%
8,192	4.23	ILU(2)	2.4	5.3	193	30	113	260	90%
12,288	5.99	ILU(1)	2.5	6.4	201	13	119	212	82%
12,288	5.99	ILU(2)	2.5	6.8	210	26	126	259	78%

we assume the base problem is defined on a base mesh with 21,927,384 elements and solved with 2,048 processor cores. Let us denote the ratio of the number of elements of the large mesh to the base mesh as "RoE", and the weak scaling parallel efficiency (EFF) is adjusted according to RoE. For example, in Table 6, RoE for 4,096 processor cores is defined as the ratio of the number of elements using 4,096 processor cores, 51,664,992, to the number of elements using 2,048 processor cores, 21,927,384. EFF is computed using the formula $(T_{np} \times np)/(T_{2,048} \times 2048) \times 1/\text{RoE}$, where T_{np} is the total compute time using np processor cores and $T_{2,048}$ is the total compute time using $2,048$ processor cores and subdomain solver ILU(1). We denote ne_{h_i}, nn_{h_i} and nu_{h_i} as the numbers of elements, nodes and unknowns for mesh Ω_{h_i}, respectively. Here Ω_{h_2} is the fine mesh, Ω_{h_1} the second coarser mesh, and Ω_{h_0} the coarsest mesh. The mesh details are shown in Table 5, and the performance results are summarized in Table 6.

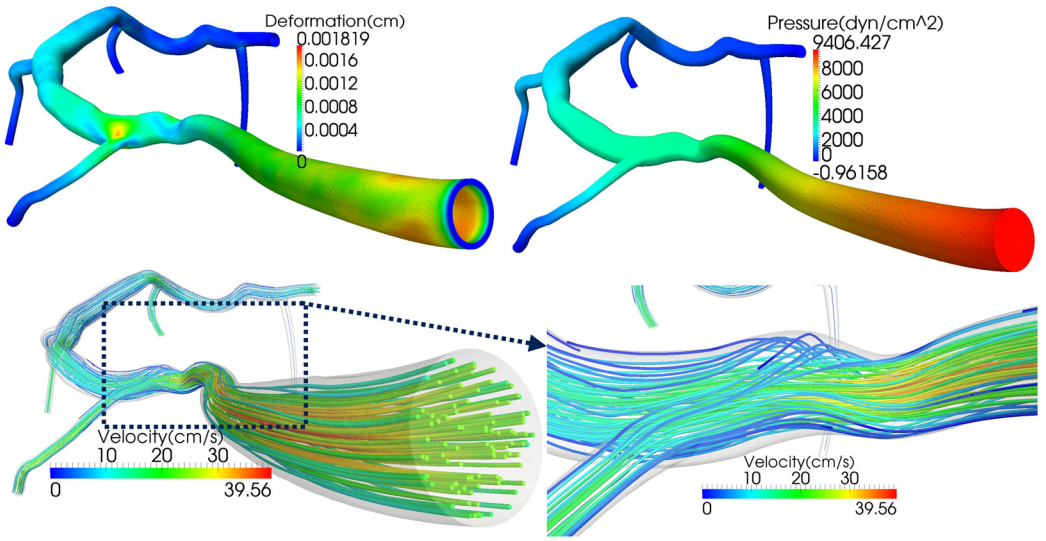

Fig. 9. Numerical solution at $t = 0.66$ s for the complex artery case. Top: arterial wall deformation and blood pressure; bottom: streamlines viewed at two different angles.

In Table 6, the numbers of Newton and fGMRES iterations stay close to constants when we increase the number of processor cores and the number of elements of the mesh. The total compute time does not increase much when more processor cores are used, which indicates that the algorithm is weakly scalable. ILU(1) is almost always better than ILU(2) in terms of the total compute time. The time for the preconditioner setup stays close to constants in most cases, and it takes about 10% of the total compute time. The weak scaling efficiency (EFF) is computed using the compute time obtained on 2,048 processor cores with ILU(1) as the base. According to EFF, the three-level Schwarz preconditioner is quite scalable in the weak sense.

5.2. A complex pulmonary artery

In this section, we consider a complex pulmonary artery, as shown in Fig. 10, from a biplane angiography image of the pulmonary artery of a patient. The density $\rho_s = 1.2$ g/cm^3, Poisson ratio $\nu_s = 0.48$, and Young's modulus $E_s = 7.5 \times 10^5$ g/(cm s^2) are used as parameters of the arterial wall. For the fluid, the density is 1.0 g/cm^3 and the viscosity is 0.035 g/(cm s). The arterial wall thickness is assumed to be 10% of the arterial diameter. To mimic the surrounding tissue effects on the arterial wall, a damping parameter $\eta = 10^7$ is used. A velocity profile with a period of 0.6 s, shown in the right figure of Fig. 12, is imposed on the fluid inlet. All fluid outlets are traction free. For the arterial wall and the fluid domain motion, the inlet and outlets are all fixed. A time step $\delta t = 10^{-3}$ is used in this test case. Numerical solution at $t = 0.66$ s is shown in Fig. 9. A sample isogeometric coarse mesh is shown in Fig. 10. For the performance study, the numerical simulation is carried out for 10 time steps with $\delta t = 10^{-3}$, that is, from 0 s to 0.01 s. The solution, shown in Fig. 9, is obtained by running the simulation for two heart beats, that is, from 0 s to 1.2 s.

(1) Different isogeometric coarse meshes. We similarly test the performance of two-level methods with two isogeometric coarse meshes. The fine mesh is denoted as Ω_{h_2}, and two isogeometric coarse meshes from fine to coarse are denoted as Ω_{h_1} and Ω_{h_0}. We combine Ω_{h_1} and Ω_{h_0} with the fine mesh Ω_{h_2} to form two two-level methods. The fine mesh has 1,508,759 vertices, 8,513,471 elements and 9,730,852 unknowns, the first coarse mesh has 764,106 vertices, 3,926,952 elements and 4,831,774 unknowns, and the second coarse has 569,960 vertices, 2,701,881 elements and 3,556,941 unknowns. Numerical results are summarized in Table 7.

In Table 7, both coarse meshes offer good scalability when using different number of processor cores. For Ω_{h_1}, the number of Newton iterations stays as a constant and the number of fGMRES iterations increases a little when we increase the number of processor cores. The numbers of linear and nonlinear iterations are kept as constants for different numbers of processor cores when we use Ω_{h_0}. The parallel speedup of using Ω_{h_0} is a little better than that obtained by using Ω_{h_1}. The performance of the algorithm with Ω_{h_0} is better than that with Ω_{h_1} in terms of the total compute time. Again, we observe that, in this test, the quality of the coarse mesh does not affect the performance because the quality of Ω_{h_0} is lower than that of Ω_{h_1}, but the performance obtained using Ω_{h_0} is better than that obtained using Ω_{h_1}.

(2) Parallel scalability with a large number of processor cores. We test the parallel scalability of a three-level Schwarz preconditioned inexact Newton–Krylov method for the complex artery. Three meshes are used to construct the three-level Schwarz preconditioner. The fine mesh has 43,423,859 vertices, 254,939,008 elements and 280,044,566 unknowns, the second coarser mesh has 721,102 vertices, 3,983,422 elements, 4,656,812 unknowns, and the coarsest mesh has 406,382 vertices, 2,041,063 elements and 2,571,102 unknowns. To maintain a linear scalability with more than 10,000 processor

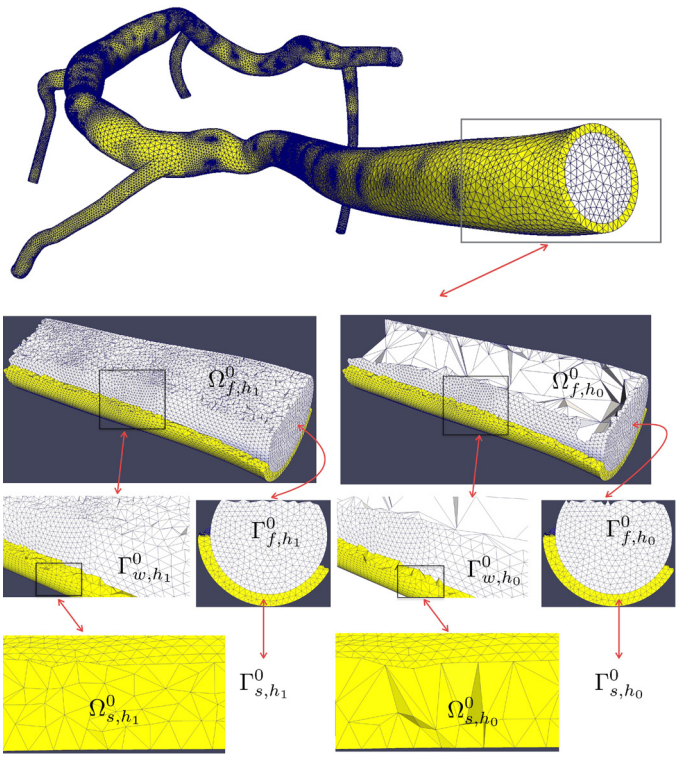

Fig. 10. Left: fine mesh, right: isogeometric coarse mesh. Here h_0 and h_1 correspond to the coarse and the fine meshes, respectively. Triangulations at Γ^0_{w,h_0}, Γ^0_{s,h_0}, Γ^0_{f,h_0} are identical to their counterparts of the fine mesh.

Table 7
Different isogeometric coarse meshes. A coupled FSI system with 9,730,852 unknowns is solved by two two-level Schwarz preconditioned inexact Newton–Krylov methods, respectively. *np* is the number of processor cores (MPI tasks), "mesh" denotes the coarse mesh for the two-level method, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "SP" is the speedup and "EFF" is the parallel efficiency.

np	mesh	NI	LI	T	PSetup	TPrec	SP	EFF
128	Ω_{h_1}	2	5.6	1043	57	765	1	100%
128	Ω_{h_0}	2	3	677	55	399	1	100%
256	Ω_{h_1}	2	6.8	616	34	476	1.69	85%
256	Ω_{h_0}	2	3.7	382	32	245	1.77	89%
512	Ω_{h_1}	2	6.9	358	21	281	2.9	73%
512	Ω_{h_0}	2	3.7	222	18	139	3.1	78%
1,024	Ω_{h_1}	2	7.7	218	18	178	4.78	60%
1,024	Ω_{h_0}	2	3.8	132	17	93	5.12	64%

cores, the two coarse meshes are much smaller than the fine mesh to save the compute time spent on the coarse levels because the coarse solves are often not scalable. 10 iterations of GMRES are used as a solver on the second coarse level. Numerical results are summarized in Table 8.

Note that, in Table 8, the parallel efficiency and speedup are calculated by comparing with the compute time obtained by using 2,048 cores so that it is easier to understand which subdomain solver is better. ILU(1) and ILU(2) do not work when we use 2,048 processor cores because there is no enough memory. ILU(0) is the best subdomain solver in this test. For all subdomain solvers, the number of Newton iterations stays as a constant when we increase the number of processor cores. The number of fGMRES iterations varies with different number of processor cores, and it impacts the efficiency and speedup but not much. For ILU(0), the efficiency is more than 77%, and at least 63% efficiency is obtained for two other subdomain solvers. The corresponding speedups and efficiencies are also plotted in Fig. 11. To further understand the three-level Schwarz method, the compute times, when using ILU(0), spent on different levels of the preconditioner, interpolation/restriction, and outer nonlinear and linear solvers are plotted in the left figure of Fig. 12. "level 2" is the fine level, "level 1" is the second coarse level, and "level 0" is the coarsest level. The compute times spent on levels 1 and 0 are

Table 8
Parallel scalability with up to 10,240 processor cores. A coupled FSI system with 280,044,566 unknowns is solved by a three-level Schwarz preconditioned inexact Newton–Krylov method. *np* is the number of processor cores (MPI tasks), "subsolve" represents the subdomain solve on the fine mesh, "NI" denotes the averaged number of Newton iterations per time step, "LI" denotes the averaged number of fGMRES iterations per Newton step, "T" is the total compute time in second for 10 time steps, "PSetup", in second, is the preconditioner setup time consisting of the construction and the ILU factorization of the subdomain matrices at all levels. "TPrec" is the time, in second, spent on the preconditioner, consisting of the setup and the application of the preconditioner on all levels. TPrec is part of T, and PSetup is part of TPrec. "MEM" is the estimated memory usage per processor core in MB, "SP" is the speedup and "EFF" is the parallel efficiency.

np	subsolve	NI	LI	T	PSetup	TPrec	MEM	SP	EFF
2,048	ILU(0)	2	11.8	1512	18	864	1099	1	100%
2,048	ILU(1)	–	–	–	–	–	–		
2,048	ILU(2)	–	–	–	–	–	–		
4,096	ILU(0)	2	8.5	698	10	363	591	2.17	108%
4,096	ILU(1)	2	11.3	907	23	570	716	1.67	83%
4,096	ILU(2)	2	5.5	793	72	460	932	1.91	95%
6,144	ILU(0)	2	10	536	8	297	431	2.82	94%
6,144	ILU(1)	2	8	545	21	307	516	2.77	92%
6,144	ILU(2)	2	6.2	599	58	363	671	2.52	84%
8,192	ILU(0)	2	14.5	491	7	313	333	3.08	77%
8,192	ILU(1)	2	12.5	526	14	347	392	2.88	72%
8,192	ILU(2)	2	7.4	489	38	313	509	3.09	77%
10,240	ILU(0)	2	8.5	325	6	180	302	4.65	93%
10,240	ILU(1)	2	14.3	481	12	333	358	3.14	63%
10,240	ILU(2)	2	7.7	414	32	268	459	3.65	73%

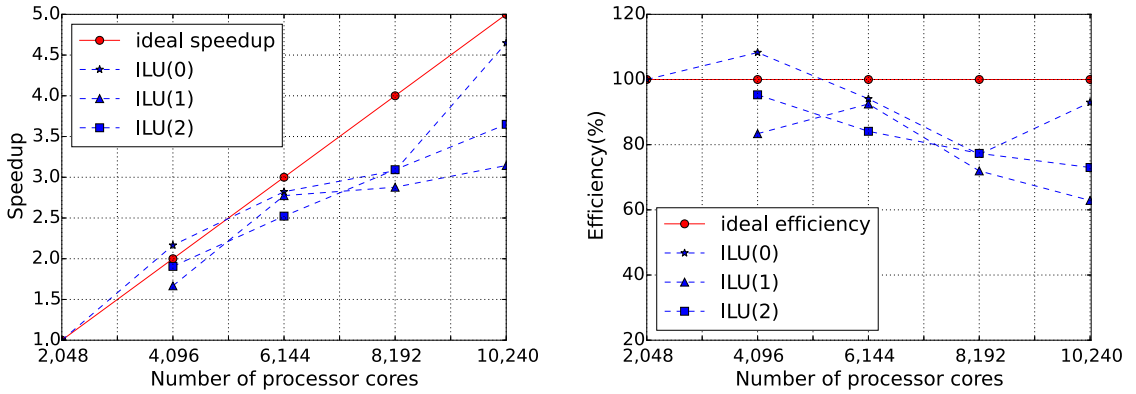

Fig. 11. Speedup and parallel efficiency using different subdomain solvers for the complex branching artery case.

not quite scalable, but they are only a small portion of the total compute time. Level 2 is scalable except at 8,192 cores, which occasionally happens for complicated problems. However, we still have a parallel efficiency of 77% with 8,092 cores. Outer solvers scale well with up to 10,240 cores. The compute time spent on the interpolation/restriction is so small that we can not clearly see it in the picture.

Finally, we mention that for practical applications of multilevel methods for unstructured mesh problems, one of the main difficulties is the generation of the multiple coarse meshes. This is done off-line using a single processor since its parallel version is yet to be developed. For our test problems, the time spent on generating two coarse meshes is less than the time spent on the generation of the fine mesh. Once the meshes are created, the Jacobian matrices have to be computed at all levels. This part of the computation is perfect for accelerators, such as GPU, but we are not exploring the possibility in this paper. Measuring the parallel performance of the proposed algorithm involves many individual components, and we only report some of them separately, including the total compute time, and the time for preconditioning, and do not individually report some of them such as Jacobian evaluations because their parallel scalability are known to be linear. Note that the time for Jacobian evaluations on all levels is included in T, and the time for Jacobian evaluations on coarse levels is included in TPrec.

6. Concluding remarks

A monolithically coupled, fully implicit Newton–Krylov solver preconditioned by a highly scalable multilevel Schwarz method was proposed for the coupled system of partial differential equations consisting of the incompressible Navier–Stokes equations for the fluid, a nonlinear elasticity equation for the solid and a linear elasticity equation for the moving fluid do-

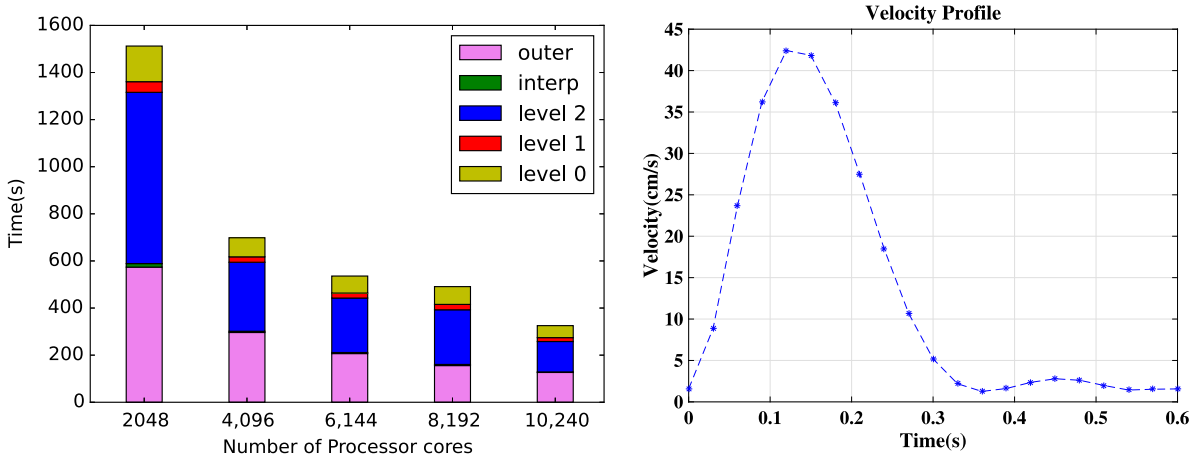

Fig. 12. Left: Compute times on different levels of the multilevel preconditioner and outer solvers when using ILU(0); right: velocity profile of the inflow. "level 0" is the coarsest level, "level 1" is the second coarse level and "level 2" is the fine level. "outer" represents the outer solver including the nonlinear and the linear solver. "interp" denotes the interpolation/restriction.

main. The coupled FSI system is discretized by a stabilized finite element method in space and a fully implicit backward difference scheme in time. The coupled algebraic system is solved by an inexact Newton method with an analytically computed Jacobian, and a Krylov subspace method together with a multilevel Schwarz preconditioner is used for the solution of the Jacobian systems. To speed up the convergence, a geometry preserving coarsening algorithm was introduced to produce several isogeometric coarse meshes that offer scalable coarse components of the multilevel preconditioner. To minimize the communication cost across different level of meshes, a hierarchical partitioning algorithm was proposed for partitioning the fine mesh and a well-matched partitioning approach was used for the partition of the coarse meshes according to the partition of the fine mesh. Numerical experiments with a patient-specific complex artery were carefully studied, and it turns out the proposed algorithm is highly scalable, in terms of both strong and weak scalabilities, with more than 10,000 processor cores for problems with hundreds of millions of unknowns on 3D unstructured meshes.

Acknowledgements

This manuscript has been authored by Battelle Energy Alliance, LLC under Contract No. DE-AC07-05ID14517 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] A. Quarteroni, M. Tuveri, A. Veneziani, Computational vascular fluid dynamics: problems, models and methods, Comput. Vis. Sci. 2 (4) (2000) 163–197.
[2] C.A. Taylor, M.T. Draney, Experimental and computational methods in cardiovascular fluid mechanics, Annu. Rev. Fluid Mech. 36 (2004) 197–231.
[3] C.A. Taylor, J.D. Humphrey, Open problems in computational vascular biomechanics: hemodynamics and arterial wall mechanics, Comput. Methods Appl. Mech. Eng. 198 (45) (2009) 3514–3523.
[4] T. Belytschko, Fluid–structure interaction, Comput. Struct. 12 (4) (1980) 459–469.
[5] E.H. Dowell, K.C. Hall, Modeling of fluid–structure interaction, Annu. Rev. Fluid Mech. 33 (1) (2001) 445–490.
[6] G. Hou, J. Wang, A. Layton, Numerical methods for fluid–structure interaction – a review, Commun. Comput. Phys. 12 (2) (2012) 337–377.
[7] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[8] C. Farhat, M. Lesoinne, P. Le Tallec, Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity, Comput. Methods Appl. Mech. Eng. 157 (1) (1998) 95–114.
[9] C. Farhat, K.G. Van der Zee, P. Geuzaine, Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity, Comput. Methods Appl. Mech. Eng. 195 (17) (2006) 1973–2001.
[10] S. Deparis, M. Discacciati, G. Fourestey, A. Quarteroni, Fluid–structure algorithms based on Steklov–Poincaré operators, Comput. Methods Appl. Mech. Eng. 195 (41) (2006) 5797–5812.
[11] G. Guidoboni, R. Glowinski, N. Cavallini, S. Canic, Stable loosely-coupled-type algorithm for fluid–structure interaction in blood flow, J. Comput. Phys. 228 (18) (2009) 6916–6937.
[12] A.T. Barker, X.-C. Cai, Scalable parallel methods for monolithic coupling in fluid–structure interaction with application to blood flow modeling, J. Comput. Phys. 229 (3) (2010) 642–659.
[13] A.T. Barker, X.-C. Cai, Two-level Newton and hybrid Schwarz preconditioners for fluid–structure interaction, SIAM J. Sci. Comput. 32 (4) (2010) 2395–2417.
[14] Y. Bazilevs, V. Calo, T.J. Hughes, Y. Zhang, Isogeometric fluid–structure interaction: theory, algorithms, and computations, Comput. Mech. 43 (1) (2008) 3–37.
[15] Y. Bazilevs, V. Calo, Y. Zhang, T.J. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Comput. Mech. 38 (4–5) (2006) 310–322.
[16] M.Á. Fernández, M. Moubachir, A Newton method using exact Jacobians for solving fluid–structure coupling, Comput. Struct. 83 (2) (2005) 127–142.

[17] Y. Wu, X.-C. Cai, A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation, J. Comput. Phys. 258 (2014) 524–537.
[18] S. Idelsohn, E. Onate, F. Del Pin, N. Calvo, Fluid–structure interaction using the particle finite element method, Comput. Methods Appl. Mech. Eng. 195 (17) (2006) 2100–2123.
[19] S.R. Idelsohn, J. Marti, A. Limache, E. Oñate, Unified Lagrangian formulation for elastic solids and incompressible fluids: application to fluid–structure interaction problems via the PFEM, Comput. Methods Appl. Mech. Eng. 197 (19) (2008) 1762–1776.
[20] L. Ge, F. Sotiropoulos, A numerical method for solving the 3D unsteady incompressible Navier–Stokes equations in curvilinear domains with complex immersed boundaries, J. Comput. Phys. 225 (2) (2007) 1782–1809.
[21] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, Appl. Mech. Rev. 56 (3) (2003) 331–347.
[22] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[23] H. Wang, J. Chessa, W.K. Liu, T. Belytschko, The immersed/fictitious element method for fluid–structure interaction: volumetric consistency, compressibility and thin members, Int. J. Numer. Methods Eng. 74 (1) (2008) 32–55.
[24] J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Int. J. Numer. Methods Eng. 46 (1) (1999) 131–150.
[25] J. Dolbow, N. Moës, T. Belytschko, An extended finite element method for modeling crack growth with frictional contact, Comput. Methods Appl. Mech. Eng. 190 (51) (2001) 6825–6846.
[26] P. Lallemand, L.-S. Luo, Lattice Boltzmann method for moving boundaries, J. Comput. Phys. 184 (2) (2003) 406–421.
[27] D. Owen, C. Leonardi, Y. Feng, An efficient framework for fluid–structure interaction using the lattice Boltzmann method and immersed moving boundaries, Int. J. Numer. Methods Eng. 87 (1–5) (2011) 66–95.
[28] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, P. Krysl, Meshless methods: an overview and recent developments, Comput. Methods Appl. Mech. Eng. 139 (1) (1996) 3–47.
[29] L.T. Zhang, G.J. Wagner, W.K. Liu, Modelling and simulation of fluid structure interaction by meshfree and FEM, Commun. Numer. Methods Eng. 19 (8) (2003) 615–621.
[30] Q. Meng, M. Berzins, Scalable large-scale fluid–structure interaction solvers in the Uintah framework via hybrid task-based parallelism algorithms, Concurr. Comput., Pract. Exp. 26 (7) (2014) 1388–1407.
[31] P. Crosetto, S. Deparis, G. Fourestey, A. Quarteroni, Parallel algorithms for fluid–structure interaction problems in haemodynamics, SIAM J. Sci. Comput. 33 (4) (2011) 1598–1622.
[32] F. Nobile, Numerical Approximation of Fluid–Structure Interaction Problems with Application to Haemodynamics, Ph.D. thesis, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 2001.
[33] C.A. Figueroa, I.E. Vignon-Clementel, K.E. Jansen, T.J. Hughes, C.A. Taylor, A coupled momentum method for modeling blood flow in three-dimensional deformable arteries, Comput. Methods Appl. Mech. Eng. 195 (41) (2006) 5685–5706.
[34] M. Zhou, O. Sahni, H.J. Kim, C.A. Figueroa, C.A. Taylor, M.S. Shephard, K.E. Jansen, Cardiovascular flow simulation at extreme scale, Comput. Mech. 46 (1) (2010) 71–82.
[35] P. Causin, J.-F. Gerbeau, F. Nobile, Added-mass effect in the design of partitioned algorithms for fluid–structure problems, Comput. Methods Appl. Mech. Eng. 194 (42) (2005) 4506–4527.
[36] S. Badia, A. Quaini, A. Quarteroni, Modular vs. non-modular preconditioners for fluid–structure systems with large added-mass effect, Comput. Methods Appl. Mech. Eng. 197 (49) (2008) 4216–4232.
[37] F. Kong, X.-C. Cai, A highly scalable multilevel Schwarz method with boundary geometry preserving coarse spaces for 3D elasticity problems on domains with complex geometry, SIAM J. Sci. Comput. 38 (2) (2016) C73–C95.
[38] F. Kong, X.-C. Cai, Scalability study of an implicit solver for coupled fluid–structure interaction problems on unstructured meshes in 3D, Int. J. High Perform. Comput. Appl. (2016), http://dx.doi.org/10.1177/1094342016646437.
[39] A.A. Johnson, T.E. Tezduyar, Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces, Comput. Methods Appl. Mech. Eng. 119 (1) (1994) 73–94.
[40] K. Stein, T. Tezduyar, R. Benney, Mesh moving techniques for fluid–structure interactions with large displacements, J. Appl. Mech. 70 (1) (2003) 58–63.
[41] Q. Zhang, T. Hisada, Analysis of fluid–structure interaction problems with structural buckling and large domain changes by ALE finite element method, Comput. Methods Appl. Mech. Eng. 190 (48) (2001) 6341–6357.
[42] P. Gamnitzer, W.A. Wall, An ALE-Chimera method for large deformation fluid structure interaction, in: ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5–8, 2006, Delft University of Technology, European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.
[43] P. Howell, G. Kozyreff, J. Ockendon, Applied Solid Mechanics, Cambridge University Press, New York, 2009.
[44] T.E. Tezduyar, S. Sathe, T. Cragin, B. Nanna, B.S. Conklin, J. Pausewang, M. Schwaab, Modelling of fluid–structure interactions with the space–time finite elements: arterial fluid mechanics, Int. J. Numer. Methods Fluids 54 (6–8) (2007) 901–922.
[45] P. Crosetto, P. Reymond, S. Deparis, D. Kontaxakis, N. Stergiopulos, A. Quarteroni, Fluid–structure interaction simulation of aortic blood flow, Comput. Fluids 43 (1) (2011) 46–57.
[46] C.A. Taylor, T.J. Hughes, C.K. Zarins, Finite element modelling of blood flow in arteries, Comput. Methods Appl. Mech. Eng. 158 (1) (1998) 155–196.
[47] C.H. Whiting, K.E. Jansen, A stabilized finite element method for the incompressible Navier–Stokes equations using a hierarchical basis, Int. J. Numer. Methods Fluids 35 (1) (2001) 93–116.
[48] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (2) (1982) 400–408.
[49] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optim. 4 (2) (1994) 393–422.
[50] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1) (1996) 16–32.
[51] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business, Media, 2006.
[52] Y. Saad, A flexible inner–outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (2) (1993) 461–469.
[53] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (2) (1999) 792–797.
[54] B. Smith, P. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 2004.
[55] A. Toselli, O. Widlund, Domain Decomposition Methods: Algorithms and Theory, Springer, Berlin, 2005.
[56] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[57] C. Yang, X.-C. Cai, Parallel multilevel methods for implicit solution of shallow water equations with nonsmooth topography on the cubed-sphere, J. Comput. Phys. 230 (7) (2011) 2523–2539.
[58] G. Karypis, K. Schloegel, ParMETIS – Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 4.0, University of Minnesota.
[59] K. Stüben, A review of algebraic multigrid, J. Comput. Appl. Math. 128 (1) (2001) 281–309.
[60] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, H. Zhang, PETSc Users Manual, Tech. Rep. ANL-95/11 – Revision 3.6, Argonne National Laboratory, 2015, http://www.mcs.anl.gov/petsc.

[61] R.D. Falgout, J.E. Jones, U.M. Yang, The design and implementation of *hypre*, a library of parallel high performance preconditioners, in: A.M. Bruaset, P. Bjorstad, A. Tveito (Eds.), Numerical Solution of Partial Differential Equations on Parallel Computers, in: Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2006, pp. 267–294.
[62] P.R. Brune, M.G. Knepley, L.R. Scott, Unstructured geometric multigrid in two and three dimensions on complex and graded meshes, SIAM J. Sci. Comput. 35 (1) (2013) A173–A191.
[63] M.F. Adams, R. Taylor, Parallel multigrid solvers for 3D unstructured finite element problems in large deformation elasticity and plasticity, Int. J. Numer. Methods Eng. 48 (8) (2000) 1241–1262.
[64] G.L. Miller, D. Talmor, S.-H. Teng, Optimal coarsening of unstructured meshes, J. Algorithms 31 (1) (1999) 29–65.
[65] C. Ollivier-Gooch, Coarsening unstructured meshes by edge contraction, Int. J. Numer. Methods Eng. 57 (3) (2003) 391–414.
[66] D.-T. Lee, B.J. Schachter, Two algorithms for constructing a Delaunay triangulation, Int. J. Comput. Inf. Sci. 9 (3) (1980) 219–242.
[67] L.P. Chew, Constrained Delaunay triangulations, Algorithmica 4 (1–4) (1989) 97–108.
[68] D.A. Field, Laplacian smoothing and Delaunay triangulations, Commun. Appl. Numer. Methods 4 (6) (1988) 709–712.
[69] L.A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, Int. J. Numer. Methods Eng. 40 (21) (1997) 3979–4002.
[70] L. Freitag, M. Jones, P. Plassmann, A parallel algorithm for mesh smoothing, SIAM J. Sci. Comput. 20 (6) (1999) 2023–2040.
[71] J. Snyman, Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-based Algorithms, Applied Optimization, vol. 97, Springer Science & Business, Media, 2005.
[72] C.F. Ollivier-Gooch, GRUMMP Version 0.5.0 User's Guide, Department of Mechanical Engineering, University of British Columbia.
[73] The CUBIT geometry and mesh generation toolkit, https://cubit.sandia.gov.