Dynamic Adaptive Moving Mesh Finite Volume Method for Navier-Stokes Equations

Alexander A. Danilov, Kirill M. Terekhov, and Yuri V. Vassilevski

Abstract This work is concerned with a parallel solution of the Navier-Stokes equations on dynamic adaptive moving meshes. The method is applied to the blood flow problem in the moving domain of the left ventricle, reconstructed from the time series of computer tomography scans. The moving mesh of the left ventricle is dynamically adapted in the areas of high vorticity to improve capturing of the flow features. For the flow approximation we use a fully implicit collocated finite volume method. The methods are implemented using functionality of the INMOST platform.

1 Introduction

The mathematical modelling of the blood flow in the heart and large vesicles becoming a clinical standard to complement Doppler sonography as a decision supporting tool for practicing cardiologists [14]. This work addresses several related challenges, such as modelling of the blood flow in the moving domain using a stable collocated finite-volume method, mesh reconstruction from time-series of computer tomography images, and an instrument that supports dynamic adaptation of the

A. A. Danilov · Y. V. Vassilevski
Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow, Russia

Moscow Institute of Physics and Technology, Dolgoprudny, Russia

Sechenov University, Moscow, Russia

Sirius University of Science and Technology, Sochi, Russia

K. M. Terekhov (✉)
Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow, Russia

Moscow Institute of Physics and Technology, Dolgoprudny, Russia

Sirius University of Science and Technology, Sochi, Russia

V. Garanzha, L. Kamenski (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 152, https://doi.org/10.1007/978-3-031-59652-0_9

moving mesh. We are concerned with a practical problem of the simulation of blood flow in the right ventricle reconstructed from time-series of computer tomography images of the heart of a patient with transposition of the great arteries.

The present work extends the collocated finite volume method, considered earlier in [22, 23, 25], to handle moving meshes. It realizes a general concept of stable flux discretization of saddle-point systems with vector of several unknowns [24]. The approach was first applied to mixed Darcy formulation in [27] and further extended to mechanics and poroelasticity [18, 20, 24, 26, 29] problems. These works demonstrated that inf-sup stability issue [10] does not affect the proposed collocated finite volume methods.

The mathematical model is built on top of INMOST, an open source library [3, 31, 32]. It provides tools for complex parallel mesh modification and balancing [17, 28], as well as tools for linear system assembly using automatic differentiation and linear system solution. There are other mesh libraries allowing for parallel mesh modification, such as Dune [1], project DuMuX [6, 9] based on Dune, STK mesh from Trilinos package [16], OpenFOAM [12], other packages for parallel mesh management are MOAB [15] and MSTK [7]. In this work we are mostly concerned with practical aspects of the solution of the Navier-Stokes problem on dynamic adaptive moving meshes using the INMOST functionality. Recently we applied INMOST to the solution of front-tracking problem using parallel adaptive meshes in [30], but we haven't studied the related performance impacts.

The article is organized as follows. In Sect. 2 we consider the INMOST functionality. In Sect. 3 we outline the construction of the sequence of moving meshes from a series of computer tomography scans. Further in Sect. 4 we are concerned with the geometrical quantities in four dimensions. In Sect. 5 we briefly discuss the finite volume method in four dimensions. At last in Sect. 7 we perform a numerical experiment.

2 INMOST Functionality

The primary functionality of the concern for the present work is the ability to handle the geometry of moving meshes, to dynamically refine and coarsen the meshes along the computations, and to perform load balancing. All of these functions are provided by the INMOST platform. The platform allows to adapt polyhedra of general shape as illustrated in Fig. 1.

The information on refined elements is stored in the hierarchy of element sets. These sets store the information necessary for cell coarsening, see Fig. 2. Distributed element sets allow to pass the information and corresponding elements across the processors to perform necessary operations.

The mesh can be modified either by low-level functions, such as removal, generation, disconnection, and reconnection of mesh elements. There are also high-level functions that allow for splitting of elements by sets of elements of lower

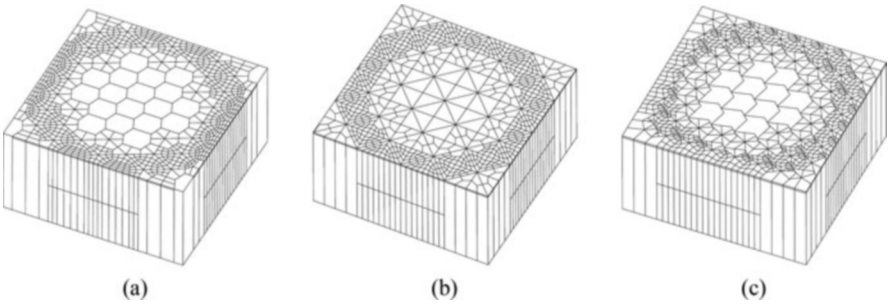


Fig. 1 General refinement of (**a**) hexagonal prisms, (**b**) triangular prisms, and (**c**) nonconvex prisms

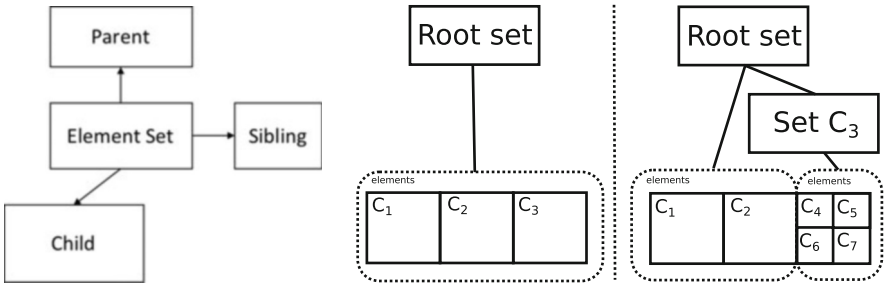
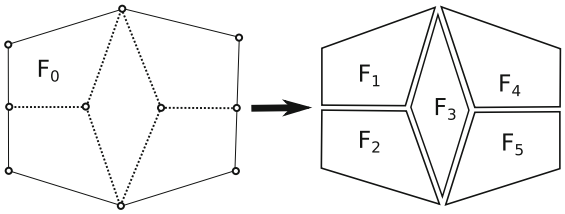


Fig. 2 Organization of sets of elements into a tree structure (left). Original unsplit mesh of three cells (middle). After splitting of cell C_3 a new set is attached to the parent (root) set (right). The set remembers all the fine cells that form the original cell C_3

Fig. 3 Separation of a face F_0 by a set of edges into faces $F_1 \ldots F_5$



dimension (see Fig. 3), union of sets of elements of the same dimension, edge collapse.

The mesh refinement is organised using high-level functionality. Each element is split by introducing a midpoint and necessary lower dimensional elements as displayed in Fig. 4a. For triangles this leads to a deterioration of the element shape as seen in Figs. 1b and 4b. To this end, we consider splitting of the triangle without midpoint. In the case of tetrahedron, prism and pyramid we also do not introduce the midpoint. The shortest inner diagonal is used to split the tetrahedron. As a result each tetrahedron is split into eight new tetrahedra. Due to face refinement

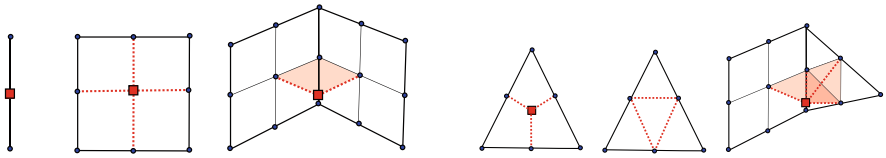


Fig. 4 (**a**) General splitting sequence with introduction of midpoints, (**b**) a special case for a triangle

adjacent coarse tetrahedra become polyhedra. In this paper only tetrahedral meshes are considered.

For load balancing, it is possible to migrate mesh elements between processors using the new partitioning. In this work we use ParMetis [8] for mesh balancing. INMOST automatically computes statuses of modified elements, equilibrates the domain boundary, and reconstructs the prescribed number of ghost layers upon mesh modification and after load balancing.

3 Constructing a Moving Mesh

The imaging corresponds to the right ventricle of a patient with transposition of the great arteries, a rare congenital defect. The motion of the ventricle is obtained from time series of computer tomography scans. The data set includes 10 contrast-enhanced CT scans with $512 \times 512 \times 304$ voxels and $0.355 \times 0.355 \times 0.5$ mm resolution. The images are resampled to $162 \times 112 \times 136$ voxels with 1 mm isotropic resolution. The right ventricle is segmented with a level-set method from the ITK-SNAP package [33].

Obtained frames are resampled to 90 frames per cardiac cycle using a cubic interpolation of the level-set function in time with the periodic conditions at the endpoints of the time interval. The boundary of the right ventricle is recovered as a zero isosurface of this interpolant. Frame 44 shows the minimum volume of the ventricle. We assume that the frames from 0 to 44 represent the systole of the cardiac cycle, and the frames from 45 to 89 represent the diastole. We approximate the average position of the valves with two static cutoff planes.

The mesh generation process was described in [4] and we refer to that paper for details. The algorithm builds a mesh in a reference domain, implicitly defined by the averaged distance function over all 90 frames. This domain is also bounded by static valve planes. A quasi-uniform unstructured tetrahedral mesh is constructed using Delaunay triangulation from the CGAL mesh library [13]. It is further improved using the aniMBA library from the Ani3D package [11].

At the next stage, the mesh is adapted to all frames from 0 to 89, which is followed by the second cycle of the mesh adaptation to ensure a smooth periodic

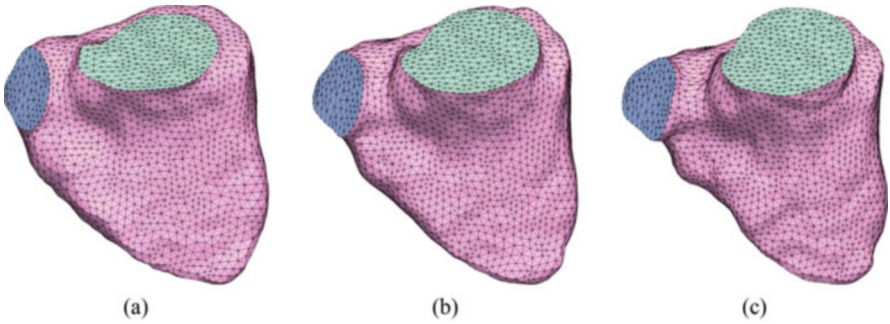


Fig. 5 The right ventricle surface mesh: (**a**) beginning of systole (frame 0), (**b**) middle of systole (frame 23), (**c**) end of systole (frame 44)

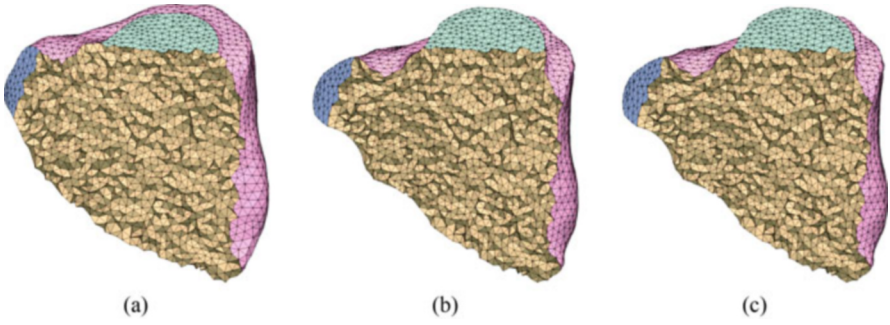


Fig. 6 The right ventricle volume mesh, the cutplane passes through the ventricle apex and the centers of the valves: (**a**) beginning of systole (frame 0), (**b**) middle of systole (frame 23), (**c**) end of systole (frame 44)

transition of the meshes from one cardiac cycle to the next one. To this end, each boundary node is shifted in the direction of the surface normal vector with the weight $1/2$ and the vector pointing to the center of the surrounding nodes with the weight $1/25$. The vertices residing on the valve planes are projected to these planes, thus ensuring that the vertices stay on valve planes (Fig. 5). At the second step, the internal nodes are shifted by a simultaneous untangling and smoothing algorithm [5].

The final result is a periodic series of 90 topologically invariant meshes with 13,222 nodes, 86,920 edges and 70,533 tetrahedra for the right ventricle; see Figs. 5 and 6. It is important to note that the reconstructed movement misses ventricle rotation and twist, that is important for the correct capturing of the flow.

4 Geometry in Four Dimensions

We assume that the geometry of Ω_n and Ω_{n+1} in temporal layers n and $(n+1)$ is known, moreover the temporal coordinate at n and $(n+1)$ is equal to t_n and t_{n+1}, respectively, as in Fig. 7. Three-dimensional cells $\omega^{n+1} = \omega(t^{n+1})$ posses collocation points with three-dimensional coordinate $\boldsymbol{x}_{\omega^{n+1}}$ and fourth coordinate t_{n+1}.

An evolution of four-dimensional cell $\omega_i(t_n, t_{n+1})$ from time moment t_n to time moment t_{n+1} is enframed by two three-dimensional cells ω_i^n and ω_i^{n+1}, i.e., it is a prism with the volume computed by $|\omega_i(t_n, t_{n+1})| = \left(|\omega_i^n| + |\omega_i^{n+1}|\right)(t_{n+1} - t_n)/2$. The volume is measured in SI by $\mathrm{m}^3\,\mathrm{s}$. By analogy the four-dimensional face $\sigma_i(t_n, t_{n+1})$ between the moments t_n and t_{n+1} is enframed with two three-dimensional faces σ_i^n and σ_i^{n+1} and has an area $|\sigma_i(t_n, t_{n+1})| = \left(|\sigma_i^n| + |\sigma_i^{n+1}|\right)(t_{n+1} - t_n)/2$, measured by $\mathrm{m}^2\,\mathrm{s}$.

Let $\boldsymbol{x}_{\sigma_i^n}, \boldsymbol{x}_{\sigma_i^{n+1}}$ correspond to three-dimensional centers, and $\boldsymbol{n}_{\sigma_i^n}, \boldsymbol{n}_{\sigma_i^{n+1}}$ to three-dimensional normals to faces $\sigma_i^n, \sigma_i^{n+1}$, respectively. We define the three-dimensional normal $\boldsymbol{n} = (\boldsymbol{n}_{\sigma_i^n} + \boldsymbol{n}_{\sigma_i^{n+1}})/2$ at the mid-point between n and $(n+1)$ layers and introduce the fourth coordinate of the normal by

$$\begin{bmatrix} \boldsymbol{n}^T & n_t \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\sigma_i^n} \\ t_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}^T & n_t \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\sigma_i^{n+1}} \\ t_{n+1} \end{bmatrix}, \tag{1}$$

that corresponds to positioning of four-dimensional coordinates of $\sigma_i^n, \sigma_i^{n+1}$ at single plane. We obtain

$$n_t = -\boldsymbol{n}^T \left(\boldsymbol{x}_{\sigma_i^{n+1}} - \boldsymbol{x}_{\sigma_i^n}\right)/(t_{n+1} - t_n) \approx -\boldsymbol{n}^T \boldsymbol{w}_{\sigma(t_n, t_{n+1})}, \tag{2}$$

where $\boldsymbol{w}_{\sigma(t_n, t_{n+1})}$ is the velocity of the face movement. Note, that the units for n_t corresponds to the velocity $\mathrm{m\,s^{-1}}$. The four-dimensional normal is given by

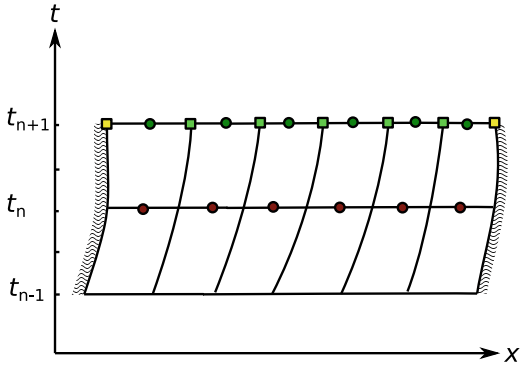


Fig. 7 Computation of the fluxes at the centers of the faces for the boundary cell $\omega(t)$ for a 1D problem. Green square—flux at the internal face, yellow square—flux at the boundary face. Green circle—collocation point at time moment t^{n+1}, red circle—collocation point at time moment t^n

$\boldsymbol{n}_t = \left[\boldsymbol{n}^T \; n_t\right]^T$. Considering that the four-dimensional coordinate is measured in $[\mathrm{m\,m\,m\,s}]^T$, and the normal in $[1\;1\;1\;\mathrm{m\,s^{-1}}]$, their scalar product is measured in m, yielding m^3 s in the divergence formula for the volume $|\omega| = \sum_\sigma |\sigma| \boldsymbol{n}^T \boldsymbol{x}$.

The outward four-dimensional normal to the face at layer $n+1$ is given by $[0\;0\;0\;1]$, and the four-dimensional face area corresponds to the volume of the cell $|\omega_i^{n+1}|$, measured in m^3. By analogy, the normal to the face at layer n is given by $[0\;0\;0\;-1]$, and it's area corresponds to the volume $|\omega_i^n|$, measured in m^3.

5 Finite Volume Method

We consider the solution of Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial \rho \boldsymbol{u}}{\partial t} + \operatorname{div}\left(\rho \boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{\tau} + p\mathbb{I}\right) = \boldsymbol{f}, & \text{in } \Omega(t), \\ \operatorname{div}(\rho \boldsymbol{u}) = 0, & \\ \boldsymbol{\alpha}(\boldsymbol{u} - \boldsymbol{w}) + \boldsymbol{\beta}(\boldsymbol{\tau} - p\mathbb{I})\boldsymbol{n} = \boldsymbol{r} & \text{on } \partial\Omega(t). \end{cases} \tag{3}$$

In (3), the domain $\Omega(t)$ changes with time, $\boldsymbol{u} = [u, v, w]^T$ is the velocity vector, where $u, v, w \in H^1(\Omega)$, and $p \in L^2(\Omega)$ is the pressure field. The spaces H^1 and L^2 are properly augmented at the boundary, $\rho = \text{const}$ is the fluid density, $\boldsymbol{\tau}$ is the stress tensor with dynamic viscosity $\mu = \text{const}$:

$$\boldsymbol{\tau} = 2\mu \mathbf{D}(\boldsymbol{u}), \quad \mathbf{D}(\boldsymbol{u}) = \frac{1}{2}\left[\boldsymbol{u}\nabla^T + \nabla \boldsymbol{u}^T\right]. \tag{4}$$

In boundary conditions $\boldsymbol{n}$ is a normal vector to $\sigma(t)$, oriented outside of the domain, $\boldsymbol{\alpha} = \alpha_\parallel \mathbb{I} + (\alpha_\perp - \alpha_\parallel)\boldsymbol{n}\boldsymbol{n}^T$ fixes the velocity at the boundary, $\boldsymbol{\beta} = \beta_\parallel \mathbb{I} + (\beta_\perp - \beta_\parallel)\boldsymbol{n}\boldsymbol{n}^T$ fixes the traction at the boundary, $\boldsymbol{w}$ is the boundary movement velocity.

Equations for $\Omega(t)$ of the system (3) can be expressed with the four-gradient as follows:

$$\begin{pmatrix} \rho \boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{\tau} + p\mathbb{I} & \rho \boldsymbol{u} \\ \rho \boldsymbol{u}^T & \end{pmatrix} \begin{pmatrix} \nabla \\ \partial_t \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}. \tag{5}$$

Applying the Ostrogradsky-Gauss theorem to the four dimensional integral of the left-hand side of (5) at every cell $\omega(t) \in \mathcal{V}(\Omega(t))$ we get:

$$\int_{\omega(t)} \begin{pmatrix} \rho \boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{\tau} + p\mathbb{I} & \rho \boldsymbol{u} \\ \rho \boldsymbol{u}^T & \end{pmatrix} \begin{pmatrix} \nabla \\ \partial_t \end{pmatrix} \mathrm{d}V(t)$$

$$\approx \sum_{\sigma(t) \in \mathcal{F}(\omega(t))} |\sigma(t)| \begin{pmatrix} \rho \boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{\tau} + p\mathbb{I} & \rho \boldsymbol{u} \\ \rho \boldsymbol{u}^T & \end{pmatrix} \begin{pmatrix} \boldsymbol{n} \\ n_t \end{pmatrix} \Bigg|_{\boldsymbol{x}_{\sigma(t)}}, \tag{6}$$

where $\mathcal{F}(\omega(t))$ is a set of faces of the cell $\omega(t)$, $[\boldsymbol{n}, n_t]^T$ is a normal vector to the four-dimensional face $\sigma(t)$, oriented outside of the cell $\omega(t)$, and $\boldsymbol{x}_{\sigma(t)}$ is a four-dimensional center and $|\sigma(t)|$ is area of the face $\sigma(t)$.

5.1 Flux Approximation

For the finite volume method it is necessary to approximate the expression under the sum in (6) at the face $\sigma(t)$ center:

$$\mathbf{F}\big|_{\boldsymbol{x}_{\sigma(t)}} = \begin{pmatrix} \rho \boldsymbol{u} n_t + \rho \boldsymbol{u}\boldsymbol{u}^T \boldsymbol{n} - \boldsymbol{\tau}\boldsymbol{n} + p\boldsymbol{n} \\ \rho \boldsymbol{n}^T \boldsymbol{u} \end{pmatrix}\Bigg|_{\boldsymbol{x}_{\sigma(t)}}, \tag{7}$$

we call this expression the coupled flux.

Let $\mathcal{V}(\Omega(t))$ represent the set of cells, covering the domain $\Omega(t)$. Each cell $\omega(t) \in \mathcal{V}(\Omega(t))$ represents a moving elementary volume. Let us introduce the pressure p_i, the velocity $\boldsymbol{u}_i$ and the four-gradient $\mathbf{G}_i$ of pressure and velocity at the four-dimensional center $\boldsymbol{x}_{\omega_i(t)}$ of each cell $\omega_i(t) \in \mathcal{V}(\Omega(t))$: $\mathbf{G}_i = \left(\begin{bmatrix}\boldsymbol{u}^T & p\end{bmatrix}^T \otimes \begin{bmatrix}\nabla^T & \partial_t\end{bmatrix}^T\right)\Big|_{\boldsymbol{x}_{\omega_i(t)}}$, here and further $A \otimes B$ is the Kronecker product. Further we consider pressure and velocity to be piecewise-continuous and their gradient piecewise-constant.

The semi-discrete approximation of the coupled flux is the following:

$$\begin{aligned}\mathbf{F}|_{\boldsymbol{x}_{\sigma(t)}} \approx (T_1 - Q_1)\begin{bmatrix}\boldsymbol{u}_1 \\ p_1\end{bmatrix} - (T_1 - D_1 - 2Q_1)\begin{bmatrix}\boldsymbol{u}_\sigma \\ p_\sigma\end{bmatrix} \\ + \left(T_1 \otimes \left(\boldsymbol{x}_{\sigma(t)} - \boldsymbol{x}_{\omega_1(t)}\right)^T - W_1\right)\mathbf{G}_1,\end{aligned} \tag{8}$$

where the matrix coefficients are given by:

$$\begin{aligned} Q_1 &= \frac{\rho}{2}\begin{bmatrix}\boldsymbol{n}^T\boldsymbol{u}_1\mathbb{I} + \boldsymbol{u}_1\boldsymbol{n}^T & \\ & 0\end{bmatrix}, \\ T_1 &= \begin{bmatrix}\left(a_1 + \frac{\mu}{r_1}\right)\left(\mathbb{I} + \boldsymbol{n}\boldsymbol{n}^T\right) & \\ & b_1\end{bmatrix}, \\ D_1 &= \begin{bmatrix}\rho n_t \mathbb{I}\,\boldsymbol{n} \\ \rho \boldsymbol{n}^T \;\; 0\end{bmatrix}, \\ W_1 &= \mu\left(\begin{bmatrix}\mathbb{I} \\ 0\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{n}^T & 0\end{bmatrix} + \begin{bmatrix}\boldsymbol{n}^T & 0\end{bmatrix} \otimes \begin{bmatrix}\mathbb{I} \\ 0\end{bmatrix}\right), \end{aligned} \tag{9}$$

where $r = \boldsymbol{n}_t \cdot \mathbf{v}$ is the distance to the face.

The coefficients a_1 and b_1 in T_1 are used to stabilize the method. In the SI system, a_1 is measured in kg m^{-2} s^{-1} and b_1 in m^{-1} s. Based on the eigenvalues analysis of $T_1 - D_1 - 2Q_1$ we use

$$a_1 = \max\left(\rho\left(\max\left(2\boldsymbol{n}^T\boldsymbol{u}_1, \boldsymbol{n}^T\boldsymbol{u}_1\right) + n_t\right) - \frac{\mu}{r_1}, \varepsilon\right) + \theta,$$

$$b_1 = \rho\left(\frac{\mu}{r_1} + a_1\right)^{-1},$$

where $\varepsilon = 10^{-3}$ is a small positive value and $\theta = \rho\sqrt{\boldsymbol{u}_1\left(\mathbb{I} - \boldsymbol{n}\boldsymbol{n}^T\right)\boldsymbol{u}_1}$.

Solving the coupled flux continuity (8) for interface pressure and velocity with a similar approximation at the neighboring cell we obtain the unique coupled flux expression:

$$\begin{aligned}\mathbf{F}|_{\boldsymbol{x}_{\sigma(t)}} \approx\ & (T_2 - D_2 - 2Q_2)(T_1 + T_2 - 2(Q_1 + Q_2))^{-1} \\ & \times\left((T_1 - Q_1)\begin{bmatrix}\boldsymbol{u}_1\\ p_1\end{bmatrix} + \left(T_1 \otimes \left(\boldsymbol{x}_{\sigma(t)} - \boldsymbol{x}_{\omega_1(t)}\right)^T - W_1\right)\mathbf{G}_1\right) \\ & - (T_1 - D_1 - 2Q_1)(T_1 + T_2 - 2(Q_1 + Q_2))^{-1} \\ & \times\left((T_2 - Q_2)\begin{bmatrix}\boldsymbol{u}_2\\ p_2\end{bmatrix} + \left(T_2 \otimes \left(\boldsymbol{x}_{\sigma(t)} - \boldsymbol{x}_{\omega_2(t)}\right)^T - W_2\right)\mathbf{G}_2\right),\end{aligned} \tag{10}$$

where $W_2 = -W_1$, $D_2 = -D_1$, and

$$Q_2 = -\frac{\rho}{2}\begin{bmatrix}\boldsymbol{n}^T\boldsymbol{u}_2\mathbb{I} + \boldsymbol{u}_2\boldsymbol{n}^T & \\ & 0\end{bmatrix}, \quad T_2 = \begin{bmatrix}\left(a_2 + \frac{\mu}{r_2}\right)\left(\mathbb{I} + \boldsymbol{n}\boldsymbol{n}^T\right) & \\ & b_2\end{bmatrix}. \tag{11}$$

Consider coupled flux approximation at the boundary face $\sigma(t) \in \mathcal{F}(\partial\Omega(t))$ adjacent to the cell $\omega_1(t) \in \mathcal{V}(\Omega(t^{n+1}))$, $\sigma(t) = \partial\Omega(t) \cap \omega_1(t)$. From the momentum conservation equation we introduce an auxiliary condition for the pressure:

$$\boldsymbol{n} \cdot \nabla p|_{\boldsymbol{x}_{\sigma(t)}} = \left(\boldsymbol{n} \cdot \boldsymbol{f} - \boldsymbol{n} \cdot \frac{\partial\rho\boldsymbol{u}}{\partial t} - \boldsymbol{n} \cdot \operatorname{div}\left(\rho\boldsymbol{u}\boldsymbol{u}^T - \boldsymbol{\tau}\right)\right)\Big|_{\boldsymbol{x}_{\sigma(t)}}. \tag{12}$$

Considering the piecewise linearity of velocity and pressure we can omit the contribution of the viscous term. The piecewise constant pressure gradient is evaluated at the cell center $\boldsymbol{x}_{\omega_1(t)}$ instead of the face center $\boldsymbol{x}_{\sigma(t)}$. Thus, we get

$$\boldsymbol{n} \cdot \nabla p|_{\boldsymbol{x}_{\omega_1(t)}} = \boldsymbol{n} \cdot \boldsymbol{f}_1 - \rho\begin{bmatrix}\boldsymbol{n}^T & 0\end{bmatrix} \otimes \begin{bmatrix}\boldsymbol{u}_1^T & 1\end{bmatrix}\mathbf{G}_1, \tag{13}$$

where $\boldsymbol{f}_1 = \boldsymbol{f}|_{\boldsymbol{x}_{\omega_1(t)}}$. Combining (13) with boundary conditions from (3) into a block system we get an expression for the unknowns at the face center:

$$\begin{bmatrix} \boldsymbol{u}_\sigma \\ p_\sigma \end{bmatrix} = (D + NT_b)^{-1}\left(R + NT_b\begin{bmatrix} \boldsymbol{u}_1 \\ p_1 \end{bmatrix} + N\Big(T_b \otimes \big(\boldsymbol{x}_{\sigma(t)} - \boldsymbol{x}_{\omega_1(t)}\big)^T - W_b\Big)\mathbf{G}_1\right), \tag{14}$$

where

$$\begin{aligned} D &= \begin{bmatrix} \boldsymbol{\alpha} & -\boldsymbol{\beta}\boldsymbol{n} \\ 0 & \end{bmatrix}, \qquad N = \begin{bmatrix} \boldsymbol{\beta} \\ 1 \end{bmatrix}, \qquad R = \begin{bmatrix} \boldsymbol{r} + \boldsymbol{\alpha}\boldsymbol{w} \\ \boldsymbol{n}\cdot\boldsymbol{f}_1 \end{bmatrix}, \\ T_b &= \begin{bmatrix} \left(a_b + \frac{\mu}{r_1}\right)\left(\mathbb{I} + \boldsymbol{n}\boldsymbol{n}^T\right) & \\ \frac{1}{r_1}\rho\left(\boldsymbol{n}^T\boldsymbol{u}_1 + n_t\right)\boldsymbol{n}^T & \frac{1}{r_1} \end{bmatrix}, \\ W_b &= \begin{bmatrix} \mu\mathbb{I} & \\ & 1 \end{bmatrix} \otimes \begin{bmatrix} \boldsymbol{n}^T & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{n}^T & 0 \end{bmatrix} \otimes \begin{bmatrix} \mu\mathbb{I} & \\ \rho\boldsymbol{u}_1^T & \rho \end{bmatrix}, \end{aligned} \tag{15}$$

where

$$a_b = \max\left(-\frac{1}{2}\left(\frac{\alpha_\perp}{\beta_\perp} + \rho\left(\boldsymbol{n}^T\boldsymbol{u}_1 + \rho n_t\right)\right) - \frac{\mu}{r_1}, \epsilon\right) + \theta$$

is a stabilizing coefficient. Using (14) in (8) we get the coupled flux at the boundary.

The coupled fluxes at the top temporal boundary $\bar{\sigma}(t) = \omega_1(t_n) \cap \omega_1(t_{n+1})$ and the bottom temporal boundary $\underline{\sigma}(t) = \omega_1(t_n) \cap \omega_1(t_{n-1})$ are:

$$\mathbf{F}|_{\bar{\sigma}(t)} = \begin{bmatrix} \rho\boldsymbol{u}_1^{n+1} \\ 0 \end{bmatrix}, \qquad \mathbf{F}|_{\underline{\sigma}(t)} = -\begin{bmatrix} \rho\boldsymbol{u}_1^{n} \\ 0 \end{bmatrix}, \tag{16}$$

with the four dimensional areas corresponding to the volumes $\omega_1(t_n)$ and $\omega_1(t_{n-1})$, respectively.

5.2 Gradient Reconstruction

We reconstruct the four-gradient $\mathbf{G}_1$ at the center of each time level $\omega_i(t) \in \Omega(t^{n+1})$. Consider a cell $\omega_1(t)$, for every other cell sharing any element $\omega_2(t) \in \mathcal{V}(\Omega(t^{n+1}) \cup \Omega(t^n))$, $\omega_1 \cap \omega_2 \neq \emptyset$, $\omega_2 \neq \omega_1$, we consider the following condition for the gradient:

$$\mathbb{I} \otimes \big(\boldsymbol{x}_{\omega_2(t)} - \boldsymbol{x}_{\omega_1(t)}\big)^T \mathbf{G}_1 = \begin{bmatrix} \boldsymbol{u}_2 \\ p_2 \end{bmatrix} - \begin{bmatrix} \boldsymbol{u}_1 \\ p_1 \end{bmatrix}. \tag{17}$$

At the boundary with prescribed conditions $\sigma(t) \in \mathcal{F}(\partial\Omega(t))$, $\sigma(t) = \partial\Omega(t) \cap \omega_1(t)$, $\omega_1(t) \in \mathcal{V}(\Omega(t^{n+1}))$ we derive the condition from (14):

$$\left(D \otimes \left(\boldsymbol{x}_{\sigma(t)} - \boldsymbol{x}_{\omega_1(t)}\right)^T + NW_b\right)\mathbf{G}_1 = R - D\begin{bmatrix}\boldsymbol{u}_1\\ p_1\end{bmatrix}. \tag{18}$$

As a result in d dimensions every condition provides $d+1$ conditions for the four-gradient $\mathbf{G}_1$, consisting of $(d+1)^2$ unknowns. It is sufficient to consider $(d+1)$ conditions. Gathering conditions (17) and (18) we obtain a system $A\mathbf{G}_1 = b$, solved with the Cholesky method: $\mathbf{G}_1 = (A^T A)^{-1} A^T b$.

5.3 Problem Solution

To avoid the solution of nonlinear problem we use the velocity from the previous time-level $\boldsymbol{u}_1^n$ in Q_1, T_1, T_b and stabilizing coefficients and $\boldsymbol{u}_2^n$ in Q_2, T_2 and stabilizing coefficients, respectively. As a result we have to assemble and solve a linear system only once. The assembly of the problem residual is performed with the following steps:

1. Reconstruct the four-gradient $\mathbf{G}_i$ with the derivatives in every cell $\omega_i(t) \in \mathcal{V}(\Omega(t^{n+1}))$ following (5.2).
2. Compute the residual $\omega_i(t) \in \mathcal{V}(\Omega(t^{n+1}))$:

$$\mathcal{R}_i = \sum_{\sigma(t)\in\mathcal{F}(\omega_i(t))} |\sigma(t)|\, \mathbf{F}|_{\boldsymbol{x}_{\sigma(t)}}, \tag{19}$$

   where the coupled flux is computed by (10) and (8)–(14).
3. For every cell $\omega_i(t) \in \mathcal{V}(\Omega(t^{n+1}))$, add to the residual $\mathcal{R}_i$ the fluxes (16) and subtract the body forces $\boldsymbol{f}_i = \boldsymbol{f}|_{\boldsymbol{x}_{\omega_i(t)}}$, multiplied by cell volume $|\omega_i(t)|$.

The matrix is assembled by the automatic differentiation of the INMOST platform. The system is further solved iteratively with the multilevel preconditioner [19, 21]. The iterative convergence tolerances are absolute $\tau_{\text{abs}} = 10^{-12}$, relative $\tau_{\text{rel}} = 10^{-18}$, dropping tolerances in the second-order incomplete factorization are $\tau_1 = 2 \times 10^{-3}$ and $\tau_2 = 5 \times 10^{-4}$, pivoting by condition estimation is $\kappa = 2$. A single overlapping layer is used for the additive Schwarz method.

6 Data Transfer

For the refinement, we use computed gradients $\mathbf{G}_1$. During the refinement step, the interpolation from a coarse cell ω_1 to fine cells ω_i is computed by

$$\begin{bmatrix} \boldsymbol{u}_i \\ p_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1 \\ p_1 \end{bmatrix} + \Theta \otimes \left(\boldsymbol{x}_{\omega_i} - \boldsymbol{x}_{\omega_0} \; 0 \right)^T \mathbf{G}_1, \tag{20}$$

where $\Theta = \operatorname{diag}\left(\theta_u, \theta_v, \theta_w, \theta_p\right)$ is chosen to limit the interpolation for all new cells ω_i:

$$\min_{\omega_j \in \mathcal{V}_n(\omega_1)} \left(\begin{bmatrix} \boldsymbol{u}_j \\ p_j \end{bmatrix} \right) \leq \begin{bmatrix} \boldsymbol{u}_1 \\ p_1 \end{bmatrix} + \Theta \otimes \left(\boldsymbol{x}_{\omega_i} - \boldsymbol{x}_{\omega_0} \; 0 \right)^T \mathbf{G}_1 \leq \max_{\omega_j \in \mathcal{V}_n(\omega_1)} \left(\begin{bmatrix} \boldsymbol{u}_i \\ p_i \end{bmatrix} \right), \tag{21}$$

here $\mathcal{V}_n(\omega_1)$ is a set of cells sharing at least a node with the cell ω_1. In (21) each component can be considered separately. The interpolation is conservative under condition $\sum_i |\omega_i| \boldsymbol{x}_i = |\omega_1| \boldsymbol{x}_1$ and is monotone due to (21). During coarsening of fine cells ω_i to a coarse cell ω_1 we use the simple averaging:

$$\begin{bmatrix} \boldsymbol{u}_1 \\ p_1 \end{bmatrix} = |\omega_1|^{-1} \sum_i |\omega_i| \begin{bmatrix} \boldsymbol{u}_i \\ p_i \end{bmatrix}, \tag{22}$$

which is both monotone and conservative.

The refinement criterion is based on the vorticity, which is computed in each cell based on $\mathbf{G}_i$:

$$\operatorname{curl}(\boldsymbol{u}_i) = \begin{bmatrix} 0\,0\,0\,0 & 0 & 0 & -1 & 0 & 0 & 1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0 & 0 & 0 & 0 & 0 & -1 & 0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0 & -1 & 0 & 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0 \end{bmatrix} \mathbf{G}_i. \tag{23}$$

The cells ω_i are marked to be refined if the $|\operatorname{curl}(\boldsymbol{u}_i)| \geq 5$. A cell is coarsened if neither of its faces indicates refinement and the coarsening level is not reached.

7 Numerical Experiment

The heart domain is measured in mm. The bounding box around the domain at the initial position has approximate sizes of $122.8\,\text{mm} \times 72.9\,\text{mm} \times 116.8\,\text{mm}$. The corresponding blood density is $\rho = 1\,060\,\text{kg}\,\text{m}^{-3} = 0.00106\,\text{g}\,\text{mm}^{-3}$ and the dynamic viscosity $\mu = 3.5\,\text{cP} = 0.0035\,\text{g}\,\text{mm}^{-1}\,\text{s}^{-1}$. We consider each mesh frame to be separated by the period $\Delta t = 0.01\,\text{s}$, which is taken as the maximal step of

the model. The initial time step is 0.0005 s, which is doubled every step until the maximal step is reached. The total simulation time is $T = 4.5$ s, which corresponds to 5 cycles.

We use two types of boundary conditions:

- Dirichlet conditions Γ_D with

$$\alpha_\perp = \alpha_\parallel = 1, \quad \beta_\perp = \beta_\parallel = 0, \quad \text{and} \quad \boldsymbol{r} = \boldsymbol{0};$$

- Directional-do-nothing conditions Γ_{DDN} with

$$\alpha_\perp = \alpha_\parallel = -\rho \left(\left| \boldsymbol{n}^T \boldsymbol{u}_1 + n_t \right| - \boldsymbol{n}^T \boldsymbol{u}_1 - n_t \right), \quad \beta_\perp = \beta_\parallel = 1, \quad \text{and} \quad \boldsymbol{r} = \boldsymbol{0}.$$

At the boundary we distinguish the systole Γ_s and the diastole Γ_d. There is a systolic phase at $T \in [0, 0.45]+0.9 \cdot i$ and a diastolic phase at $T \in [0.45, 0.9]+0.9 \cdot i$, where $i \in \mathbb{N}_0$. In the systolic phase the systole Γ_s is closed (Γ_D) and the diastole Γ_d is open (Γ_{DDN}). In the diastolic phase the systole Γ_s is open (Γ_{DDN}) and the diastole Γ_d is closed (Γ_D). At the rest of the boundary $\partial\Omega \setminus \Gamma_s \cup \Gamma_d$ the Dirichlet conditions Γ_D are prescribed.

The initial velocity and pressure are zero.

We run the simulation in parallel on INM RAS cluster [2] using 6 nodes with 40 cores totalling 240 processes.

The solution at various times over the heart cycle is depicted in Fig. 8. The change of velocity and pressure with time is illustrated in Fig. 9.

The corresponding mesh that emphasizes the refinement regions is found in Fig. 10. The evolution of the number of cells is displayed in Fig. 11a. The distribution of the mesh among the processors is illustrated in Fig. 12. The mesh is repartitioned by ParMetis after each refinement and coarsening step. The evolution of the number of cells and the balance ratio (ratio of maximal local elements to minimum elements) is given in Fig. 11b. It shows that at some of the steps the

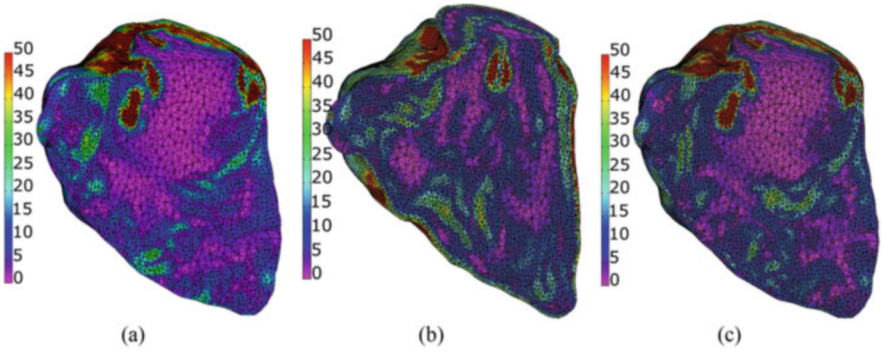


Fig. 8 Cutaway of the mesh colored in $|\mathrm{curl}(\boldsymbol{u})|$ at (**a**) $T = 3.6$, (**b**) $T = 4.05$, (**c**) $T = 4.5$

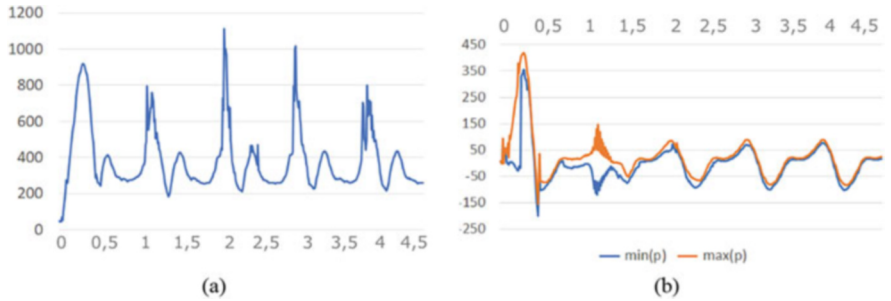


Fig. 9 Change over time of (**a**) the maximal absolute velocity (mm s^{-1}) and (**b**) the minimal and maximal pressure

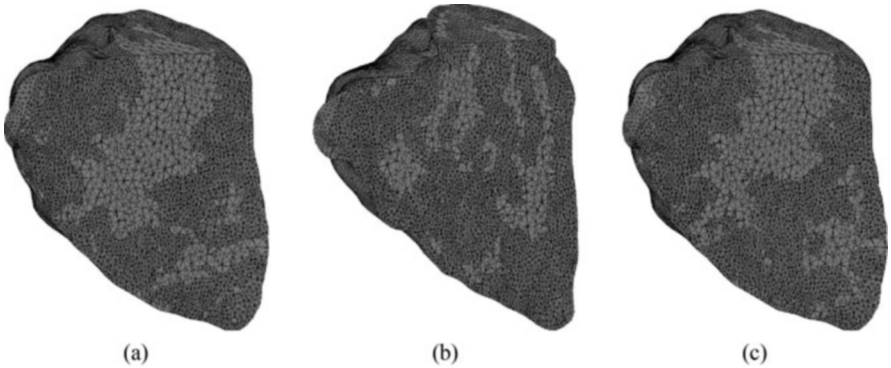


Fig. 10 Cutaway of the mesh at (**a**) $T = 3.6$, (**b**) $T = 4.05$, (**c**) $T = 4.5$

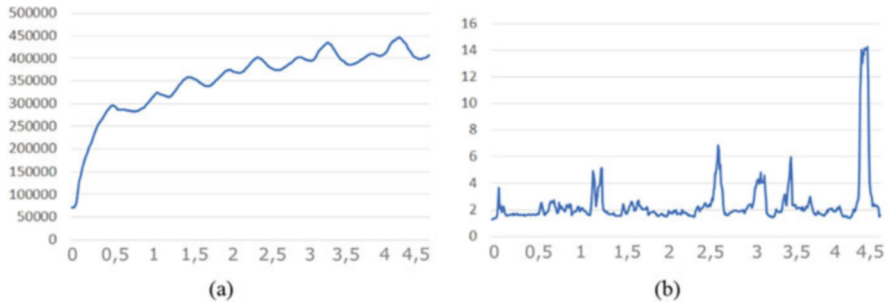


Fig. 11 Change over time of (**a**) the number of cells and (**b**) the mesh balance ratio

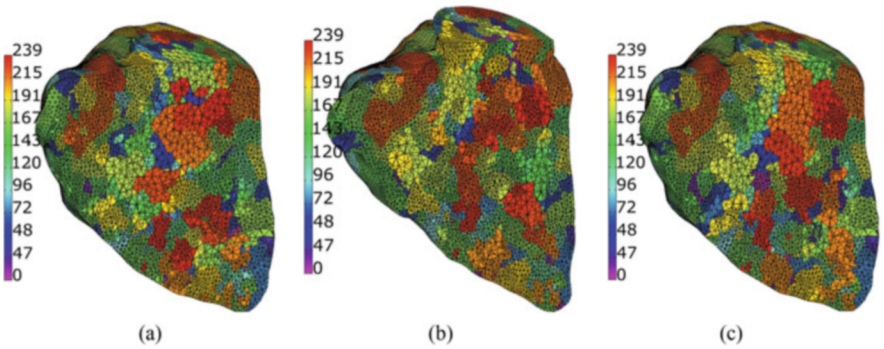


Fig. 12 Cutaway of the mesh colored by partition number at (**a**) $T = 3.6$, (**b**) $T = 4.05$, (**c**) $T = 4.5$

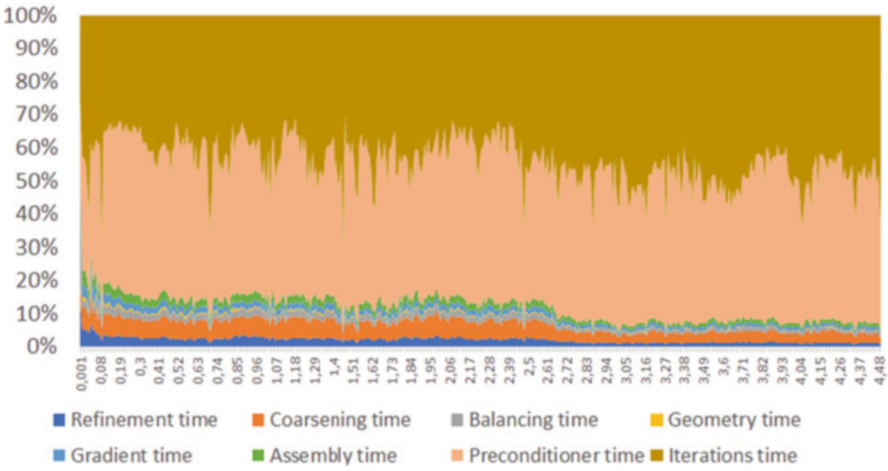


Fig. 13 Percentage of time consumed by each step of the model

disbalance is significant, which may negatively impact linear solver efficiency and convergence.

A single step of the model is split into steps:

- refinement and balancing;
- geometry and gradient computation, system assembly;
- system preconditioning and iterative solution;
- coarsening and balancing.

The cost of individual steps in percentage of the total step time is given in Fig. 13. Although the time for mesh adaptation and balancing is considerable with respect to the system assembly time, it is evident that the total solution time is dominated by the linear system solution. The ratio between the time consumed by the preconditioner and the iterative solution is given in Fig. 14a, which shows that

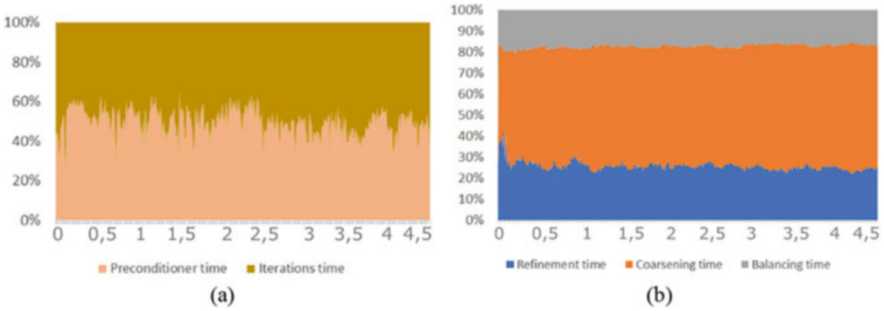


Fig. 14 Percentage of time consumed by (**a**) system preconditioning and iterative system solution and (**b**) mesh adaptation and balancing

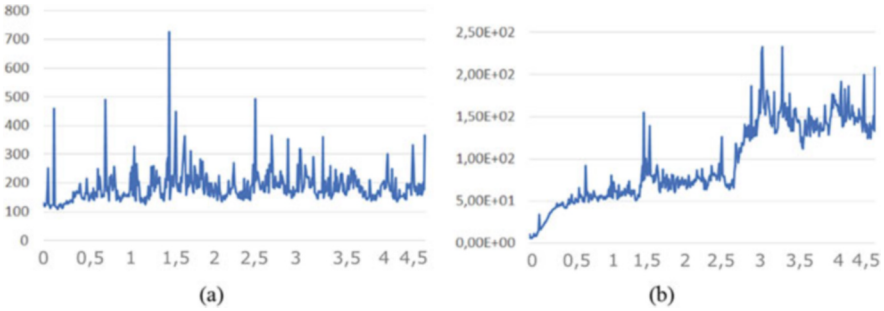


Fig. 15 Change over time of (**a**) linear iterations required to solve the system and (**b**) time in seconds consumed per time step

the costs are comparable. In the adaptation step, the coarsening step dominates the overall time and the mesh balancing is negligible as seen in Fig. 14b. The number of linear iterations is given in Fig. 15a. On average, the number of linear iterations stays the same despite the growth in the number of cells. This is due to the adaptive nature of the solver. However, the solution cost rises significantly with the number of cells. The time needed to evaluate a single time step is given in Fig. 15b.

8 Conclusion

In this work we have considered the solution of Navier-Stokes equations on a dynamic adaptive moving mesh. It shows the ability of the collocated finite-volume method to handle complex general meshes with hanging elements as well as the ability of the INMOST platform to manage such meshes in parallel. The presented results indicate that the efficiency bottleneck is in the linear solver. This calls for a more efficient multigrid approach. Here we focused on the technical aspects of the

simulation. The detailed analysis of the convergence of the method will be presented in forthcoming works.

Acknowledgments Supported by the Moscow Center of Fundamental and Applied Mathematics (Agreement 075-15-2022-286 with the Ministry of Education and Science of the Russian Federation).

References

1. Bastian, P., Blatt, M., Dedner, A., Dreier, N.A., Engwer, C., Fritze, R., Gräser, C., Grüninger, C., Kempf, D., Klöfkorn, R., et al.: The dune framework: basic concepts and recent developments. Comput. Math. Appl. **81**, 75–112 (2021)
2. Danilov, A.: INM RAS cluster. Accessed 22 Aug 2022. https://cluster2.inm.ras.ru
3. Danilov, A.A., Terekhov, K.M., Konshin, I.N., Vassilevski, Y.V.: Parallel software platform INMOST: a framework for numerical modeling. Supercomput. Front. Innov. **2**(4), 55–66 (2015)
4. Danilov, A., Lozovskiy, A., Olshanskii, M., Vassilevski, Y.: A finite element method for the Navier-Stokes equations in moving domain with application to hemodynamics of the left ventricle. Russ. J. Numer. Anal. Math. Model. **32**(4), 225–236 (2017)
5. Escobar, J., Rodríguez, E., Montenegro, R., Montero, G., González-Yuste, J.: Simultaneous untangling and smoothing of tetrahedral meshes. Comput. Methods Appl. Mech. Eng. **192**(25), 2775–2787 (2003)
6. Flemisch, B., Darcis, M., Erbertseder, K., Faigle, B., Lauser, A., Mosthaf, K., Müthing, S., Nuske, P., Tatomir, A., Wolff, M., et al.: Dumux: Dune for multi-{phase, component, scale, physics,...} flow and transport in porous media. Adv. Water Resour. **34**(9), 1102–1112 (2011)
7. Garimella, R.V.: MSTK - a flexible infrastructure library for developing mesh based applications. In: International Meshing Roundtable, pp. 213–220. Citeseer (2004)
8. Karypis, G., Schloegel, K., Kumar, V.: Parmetis. Parallel graph partitioning and sparse matrix ordering library. Version 2 (2003)
9. Koch, T., Gläser, D., Weishaupt, K., Ackermann, S., Beck, M., Becker, B., Burbulla, S., Class, H., Coltman, E., Emmert, S., et al.: Dumux 3–an open-source simulator for solving flow and transport problems in porous media with a focus on model coupling. Comput. Math. Appl. **81**, 423–443 (2021)
10. Ladyzhenskaya, O.: The Mathematical Theory of Viscous Incompressible Flow, vol. 12. Gordon & Breach, New York (1969)
11. Lipnikov, K., Vassilevski, Y., Danilov, A., et al.: Advanced numerical instruments 3d. https://sourceforge.net/projects/ani3d
12. Rettenmaier, D., Deising, D., Ouedraogo, Y., Gjonaj, E., De Gersem, H., Bothe, D., Tropea, C., Marschall, H.: Load balanced 2d and 3d adaptive mesh refinement in openfoam. SoftwareX **10**, 100317 (2019)
13. Rineau, L., Yvinec, M.: A generic software design for Delaunay refinement meshing. Comput. Geom. **38**(1–2), 100–110 (2007)
14. Rodriguez Muñoz, D., et al.: Intracardiac flow visualization: current status and future directions. Eur. Heart J. Cardiovasc. Imaging **14**(11), 1029–1038 (2013)
15. Tautges, T.J., Ernst, C., Stimpson, C., Meyers, R.J., Merkley, K.: MOAB: a mesh-oriented database. Tech. rep. SAND2004-1592, Sandia National Laboratories (2004)
16. Trilinos Project Team, T.: The Trilinos Project Website
17. Terekhov, K.: Parallel dynamic mesh adaptation within INMOST platform. In: Russian Supercomputing Days, pp. 313–326. Springer, Berlin (2019)

18. Terekhov, K.: Cell-centered finite-volume method for heterogeneous anisotropic poromechanics problem. J. Comput. Appl. Math. **365**(112357) (2020)
19. Terekhov, K.: Parallel multilevel linear solver within inmost platform. In: Russian Supercomputing Days, pp. 297–309. Springer, Berlin (2020)
20. Terekhov, K.M.: Multi-physics flux coupling for hydraulic fracturing modelling within inmost platform. Russ. J. Numer. Anal. Math. Model. **35**(4), 223–237 (2020)
21. Terekhov, K.: Greedy dissection method for shared parallelism in incomplete factorization within inmost platform. In: Russian Supercomputing Days, pp. 87–101. Springer, Berlin (2021)
22. Terekhov, K.M.: Collocated finite-volume method for the incompressible Navier–Stokes problem. J. Numer. Math. **29**(1), 63–79 (2021)
23. Terekhov, K.M.: Fully-implicit collocated finite-volume method for the unsteady incompressible Navier–Stokes problem. In: Numerical Geometry, Grid Generation and Scientific Computing. Lect. Notes Comput. Sci. Eng., vol. 143, pp. 361–374. Springer, Cham (2021)
24. Terekhov, K.M.: General finite-volume framework for saddle-point problems of various physics. Russ. J. Numer. Anal. Math. Model. **36**(6), 359–379 (2021)
25. Terekhov, K.M.: Pressure boundary conditions in the collocated finite-volume method for the steady Navier–Stokes equations. Comput. Math. Math. Phys. **62**(8), 1343–1353 (2022)
26. Terekhov, K., Tchelepi, H.: Cell-centered finite-volume method for elastic deformation of heterogeneous media with full-tensor properties. J. Comput. Appl. Math. **364**(112331) (2020)
27. Terekhov, K., Vassilevski, Y.: Finite volume method for coupled subsurface flow problems, I: Darcy problem. J. Comput. Phys. **395**, 298–306 (2019)
28. Terekhov, K., Vassilevski, Y.: Mesh modification and adaptation within inmost programming platform. In: Numerical Geometry, Grid Generation and Scientific Computing. Lect. Notes Comput. Sci. Eng. 131, pp. 243–255. Springer, Cham (2019)
29. Terekhov, K.M., Vassilevski, Y.V.: Finite volume method for coupled subsurface flow problems, II: Poroelasticity. J. Comput. Phys. **462**, 111225 (2022)
30. Vassilevski, Y.V., Terekhov, K.: Nonlinear finite volume method for the interface advection-compression problem on unstructured adaptive meshes. Comput. Math. Math. Phys. **62**(7), 1041–1058 (2022)
31. Vassilevski, Y., Konshin, I., Kopytov, G., Terekhov, K.: INMOST - A Software Platform and a Graphical Environment for Development of Parallel Numerical Models on General Meshes. Moscow State Univ. Publ., Moscow (2013)
32. Vassilevski, Y., Terekhov, K., Nikitin, K., Kapyrin, I.: Parallel Finite Volume Computation on General Meshes. Springer, Berlin (2020)
33. Yushkevich, P.A., Piven, J., Hazlett, H.C., Smith, R.G., Ho, S., Gee, J.C., Gerig, G.: User-guided 3D active contour segmentation of anatomical structures: significantly improved efficiency and reliability. NeuroImage **31**(3), 1116–1128 (2006)