WAIT: Workshop on the Advances in Information Technology, Budapest, 2018

Finite volume blood flow simulation for highly deformable boundaries

Ágota Kacsó, László Szécsi, Márton Tóth, Balázs Benyó and Tamás Umenhoffer

Department of Control Engineering and Information Technology, Budapest University of Technology and Economics

Abstract

We present the theoretical background and implementation results of a finite volume flow simulation technique that we have adapted to the GPU. The solution works on unstructured grids, and can be extended to moving grids under large deformations. Thus, it is a suitable tool for the simulation of blood flow in heart valves, especially in the context of evaluating strategies for heart valve leaflet alignment in aortic root replacement surgery.

Categories and Subject Descriptors (according to ACM CCS): J.2 [Phyiscal Sciences and Engineering]:

1. Introduction

Predictive simulation of blood flow in the human body can be used to aid the planning of surgeries modifying elements of the circulation organs. Heart surgery, in particular, creates a new, complex dynamic system. The full simulation of this system, while extremely challenging, can provide the means for predicting the outcomes of surgical options, leading to results closer to the ideal. Turbulent blood flow in the heart, and around the heart valves in particular, is extremely difficult to predict, but it may influence the efficacy of these systems. If heart valve leaflets do not fully unfold or there remain gaps between them, blood may leak back, reducing the efficiency of the circulation.

The simulation of cardiac operation would involve modeling its musculature, its elastic tissues, and the blood flow. In this paper, we focus on the blood flow simulation problem, but we keep in mind that elastic elements will also contribute. At the very least, this means that the boundary conditions of the flow simulation may vary wildly in time, producing extremely strong distortions of the geometry. We do not consider the simulation of the elastic elements in any detail, as they are discussed elsewhere[10]. Please note that, in case of a dynamic simulation, the result of the flow computation must also act as an input to the elastic system, producing a two-phase algorithm where the two sub-systems exchange boundary conditions in every time step. While this is a widespread approach, an integrated simulation of the elastic and fluid components could probably reach higher levels of accuracy. Therefore, we keep the possibility of such an extension open in our approach.

Blood itself is slightly compressible, because of the elastic red blood cells suspended in it, and it is known to be a shear thinning, non-Newtonian fluid[1]. However, these properties are only significant at the scale of arteries in the micrometer range. Thus, for flow in the heart, incompressibility and Newtonian behavior are solid assumptions.

2. Motivation for a finite volume approach

Eulerian and Lagrangian approaches to fluid simulation both have advantages and drawbacks, especially in computations concerning interactions with elastic materials[7]. We cannot dispense with the ability of Lagrangian methods to handle large deformations without issues. Eulerian methods, on the other hand, may reach higher precision, and incompressibility is easier to ensure.

We considered Eulerian simulation on a regular grid, but the geometry of heart valve leaflets (the thin membrane and potentially very narrow gaps) would have required an extremely high resolution that would have not been acceptable in terms of either memory consumption or computational load. Therefore, the problem must be addressed using an unstructured grid. This allows the use of high resolution near narrow gaps. Furthermore, boundary surfaces on both sides of heart valve leaflets can be handled easily, as spatial proximity does not have to correspond to grid adjacency, as it is the case with a regular grid.

We narrowed down the set of flow simulation approaches according to additional criteria. For one part, we needed a method to simulate unstable, incompressible flow. On the other part, we were looking for a method able to handle boundary surfaces with large time-dependent translations. These properties point to an ALE (Arbitrary Lagrangian Eulerian) approach[7]. These essentialy carry out computations on elements not strictly bound to either the spatial locations or the simulated medium. In practice, this means finite element methods where the grid itself is strongly moving, and its topology may also change in time. A subgroup of finite element methods are finite volume formulations, often employed in flow simulation. These are advantageous in that they work well for unstructured grids, and that they are conservative, i.e. they ensure the conservation of mass as well as incompressibility to a high degree. Among finite volume methods, we regarded node-based solutions to be more fitting. These store flow variables at nodes, not at elements or sample points within elements. A different name for the same approach is the *control volume method*. Such a formulation makes it possible to rebuild the grid even after severe deformations. Even though the computations are very different, one can draw parallels to the Smoothed Particle Hydrodynamics[6] method, the difference being that here we construct a momentary grid to compute effects on the particles (i.e. the nodes of the grid). This obviously needs more computing power, but allows adherence to more rigorous criteria on accuracy. The commercial software package Ansys Fluent[5] is also using a solution for blood flow simulation that falls in this category.

According to the above considerations, we based or solution on a node-based finite volume approach.

The theoretical background follows the works of Zhou and Forhad[12] in general. Spatial subdivision with GPU support is implemented using Gdel[2].

3. The base equation

The Navier-Stokes equations are written for the incompressible case in dimensionless form (normed to some characteristic length and free velocity) as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{j}_c = \nabla \cdot \mathbf{j}_v, \qquad (1)$$

where $\mathbf{U} = \begin{bmatrix} u \\ v \end{bmatrix}$ is flow velocity. Convective flux $\mathbf{j}_c$ is a vector of vectors

$$\mathbf{j}_c = \begin{bmatrix} u\mathbf{U} \\ v\mathbf{U} \end{bmatrix}.$$

Viscous flux $\mathbf{j}_v$ is also a vector of vectors:

$$\mathbf{j}_v = \frac{1}{\text{Re}} \begin{bmatrix} \nabla u \\ \nabla v \end{bmatrix},$$

where Re is the Reynolds number and ∇u, ∇v are the gradients of velocity components u and v, respectively.

The method of artificial compressibility[3] extends the above by adding pressure into the vector of flow variables. The artificial compressibility β and pseudo-time τ are also introduced. We allow compression of the fluid, but by advancing the pseudo-time the solution converges to the incompressible one. Thus stepping the pseudo-time means a kind of iteration, and the artificial compressibility is a kind of relaxation parameter.

The equations are modified as follows:

$$\frac{\partial \mathbf{W}}{\partial \tau} + \mathbf{K}\frac{\partial \mathbf{W}}{\partial t} + \nabla \cdot \mathbf{F}_c = \nabla \cdot \mathbf{F}_v, \qquad (2)$$

where

$$W = \begin{bmatrix} p \\ u \\ v \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Convective flux is augmented with the artificial compressibility, and also with the effect of pressure differences:

$$\mathbf{F}_c = \begin{bmatrix} \beta\mathbf{U} \\ u\mathbf{U} + \begin{bmatrix} p/\rho \\ 0 \end{bmatrix} \\ v\mathbf{U} + \begin{bmatrix} 0 \\ p/\rho \end{bmatrix} \end{bmatrix}. \qquad (3)$$

In the Navier-Stokes equation this flux appears behind the divergence operator. Pressure appearing in the fluxes of velocity components thus introduces the divergence of pressure into the equation, as we would expect in the compressible case.

Viscosity does not affect pressure, thus:

$$\mathbf{F}_v = \frac{1}{\text{Re}} \begin{bmatrix} 0 \\ \nabla u \\ \nabla v \end{bmatrix}. \qquad (4)$$

4. Computational method

Equation 2 is discretized over an irregular triangle or tetrahedron grid. Control volumes are polygons (or polyhedra) centered around grid nodes, with some of their vertices located at the centroids of the triangles (or tetrahedra). Other vertices are at the midpoints of the edges adjacent to the node (Figure 1). In case of a central Voronoi grid the control volumes coincide with Voronoi cells.

Equation 2, integrated over the control volumes, can be written as:

$$\frac{\partial \mathbf{W}_P \Delta S_{cv}}{\partial \tau} + \mathbf{K}\frac{\partial \mathbf{W}_P \Delta S_{cv}}{\partial t} + \int_{L_{cv}} \mathbf{F}_c \cdot \mathbf{n} dl = \int_{L_{cv}} \mathbf{F}_v \cdot \mathbf{n} dl, \qquad (5)$$

where $\mathbf{W}_P$ is the average value of flow variables in the control volume, ΔS_{cv} is the volume of the control volume, and the volumetric integrals of the divergences have been replaced by surface integrals. The surface normal at l is denoted by $\mathbf{n}$.

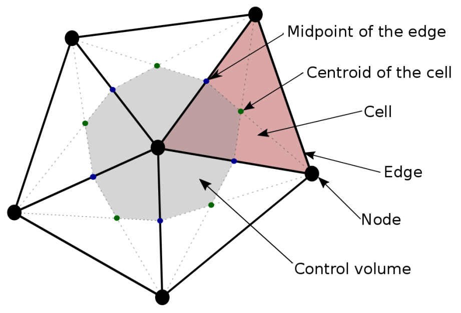

Figure 1: *The control volume of a node.*

When computing surface integrals over the control volumes' surface, sets of edges (or faces) in regions where the integrand is assumed to be constant can be replaced by edges (or faces) with different topology but identical endpoints (or boundaries). In particular, when integrating a quantity which is a per-cell constant, the centroids can be omitted (figure 2), and integration can be performed over the segment (or face) connecting the edge midpoints. Conversely, when integrating a quantity that is given as a constant along an edge, the midpoints should be dropped, and the integration is performed over the edges (or faces) of the dual grid (figure 3).

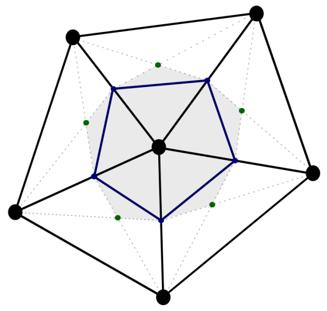

Figure 2: *The surface of the control volume when integrating a per-cell constant quantity.*

In order to compute the surface integral of viscous flux, we sum up the fluxes computed in the cells (the triangles or tetrahedra) for the faces of the control volume with the centroids dropped.

$$\int_{L_{\mathrm{cv}}} \mathbf{F}_{\mathrm{v}} \cdot \mathbf{n} \mathrm{d}l = \sum_{i=1}^{\mathrm{ncell}} \mathbf{F}_{\mathrm{v}i} \cdot \Delta \mathbf{l}_{\mathrm{C}i}, \tag{6}$$

where $\mathbf{F}_{\mathrm{v}i}$ is the flux computed in the i-th cell adjoint to the central node of the control volume, and $\Delta\mathbf{l}_{\mathrm{C}i}$ is the normal of control volume face in the same cell, weighted by the area of the face.

Thus, we sum the flows through the faces. In order to

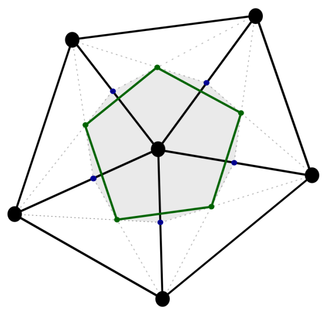

Figure 3: *The surface of the control volume when integrating a quantity that is constant along an edge.*

compute flux $\mathbf{F}_{\mathrm{v}i}$, we need to evaluate the gradients of the velocity components according to equation 4, in every cell. For any Φ flow variable (may that be the pressure, or, as in equation 4, a component of velocity) we can use the Kelvin-Stokes theorem to compute the gradient as follows:

$$\nabla\Phi = -\frac{1}{4}\frac{\sum_{i=1}^{4}\Phi_i \mathbf{l}_i}{V}, \tag{7}$$

where Φ_i is the value of the flow variable at the i-th node, $\mathbf{l}_i$ is the normal times area of the cell's opposing face, and V is the cell's volume.

We divide the surface of the control volumes differently from the above to evaluate surface integral of the convective flux. The fluxes computed for the edges connecting grid nodes are integrated over the surfaces spanned by centroids of cells adjacent to the edge. Let us call these *edge cross sections*. Zhou and Forhad[12] proposed to first evaluate the fluxes projected to the edges connecting the nodes, then re-project these per-edge fluxes (scalar for each flow variable) onto the edge cross sections. We find this is only appropriate for central Voronoi grids. Otherwise, the quantity projected twice is not identical to the one we would obtain by a single projection. Therefore, we compute the flow through edge cross sections directly. Thus:

$$\int_{L_{\mathrm{cv}}} \mathbf{F}_{\mathrm{c}} \cdot \mathbf{n} \mathrm{d}l = \sum_{k=1}^{\mathrm{nedge}} (Q_{\mathrm{c}})_{ij}^{k}, \tag{8}$$

where $(Q_{\mathrm{c}})_{ij}^{k}$ is the flow through the edge cross section of edge k, the endpoints of which are nodes i and j.

At nodes, the convective flux, and its derivative according to W, can easily be computed using formula 3. Using fluxes and flow variables at nodes the flow across an edge cross section can be approximated as follows:

$$Q_{ij} \approx \frac{1}{2}\left[\mathbf{F}_i \cdot \mathbf{n}_{ij} + \mathbf{F}_j \cdot \mathbf{n}_{ij} - |\lambda_{ij}|(W_i - W_j)\right], \tag{9}$$

where $\mathbf{n}_{ij}$ is the normal times the area of the edge cross section, and the spectral radius λ_{ij} is computed as:

$$\lambda_{ij} = \mathbf{U} \cdot \mathbf{n}_{ij} + \sqrt{(\mathbf{U} \cdot \mathbf{n}_{ij})^2 + \beta^2}.$$

Later on, we also need the derivatives of the edge cross section flows:

$$\frac{\partial Q_{ij}}{\partial W_i} = \frac{1}{2}\left[\frac{\partial \mathbf{F}_i}{\partial W_i} \cdot \mathbf{n}_{ij} - |\lambda_{ij}|\right]. \qquad (10)$$

With the above, residuum R can be computed:

$$R(W_P) = \sum_{k=1}^{\text{nedge}} (Q_\text{c})_{ij}^k - \sum_{i=1}^{\text{ncell}} \mathbf{F}_{\text{v}i} \cdot \Delta \mathbf{l}_{\text{C}i}.$$

Discretization with respect to physical time is performed using a second order implicit scheme. Considering the change of flow variables in physical time, residuum in time step $(n+1)$ is:

$$\tilde{R}(W_P^{n+1}) = R(W_P^{n+1})$$

$$+ \frac{1.5 W_P^{n+1} S_\text{cv}^{n+1} - 2 W_P^n S_\text{cv}^n + 0.5 W_P^{n-1} S_\text{cv}^{n-1}}{\text{d}t}.$$

Discretization according to the pseudo-time is done with the explicit fifth-order Runge-Kutta method. We do not reproduce the derivation[12] here, but the residuum is obtained as:

$$\tilde{\tilde{R}}_i^{m+1,n} = \frac{\tilde{R}_i^{m+1,n}}{\frac{\Delta t + 1.5\Delta\tau}{\Delta t} - \frac{\partial R_i}{\partial W_i}\frac{\Delta\tau}{\Delta S_{\text{cv}i}}},$$

where derivatives $\frac{\partial R_i}{\partial W_i}$ are obtained by formal derivation of formula 8, and the derivatives in the summation are obtained by evaluation of formula 10.

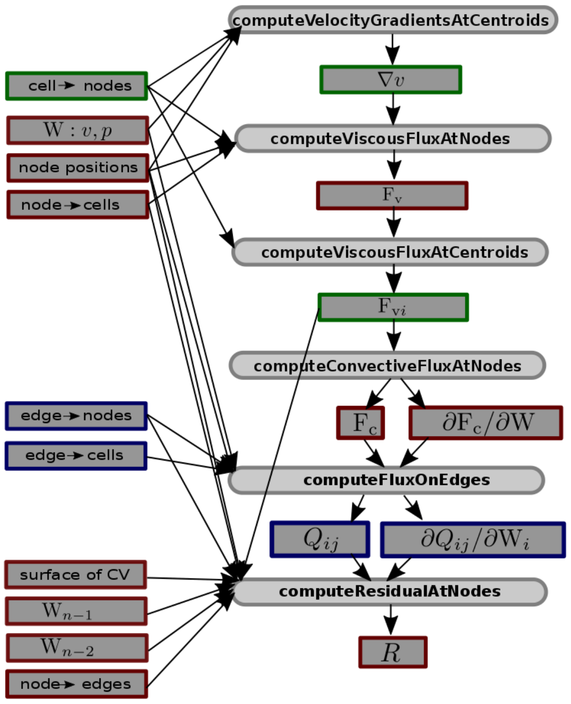
computeVelocityGradientsAtCentroids
∇v
cell→ nodes
W : v, p
node positions
node→cells
computeViscousFluxAtNodes
$\mathrm{F_v}$
computeViscousFluxAtCentroids
$\mathrm{F}_{\mathrm{v}i}$
computeConvectiveFluxAtNodes
edge→nodes
edge→cells
$\mathrm{F_c}$
$\partial\mathrm{F_c}/\partial\mathrm{W}$
computeFluxOnEdges
Q_{ij}
$\partial Q_{ij}/\partial \mathrm{W}_i$
surface of CV
W_{n-1}
W_{n-2}
node→ edges
computeResidualAtNodes
R

Figure 4: *CUDA kernels and data buffers. Red, green, and blue edges indicate per node, per cell, and per edge data, respectively. Arrows in buffer labels indicate adjacency list buffers that describe grid topology.*

5. Implementation

We have implemented the method in C++. Computations were implemented both using the CPU and the GPU (with CUDA). For this, grid connectivity was encoded in data buffers, computations were decomposed to parallel steps, storing their inputs and outputs in data buffers (figure 4). The method takes m steps (20 in our tests) of pseudo-time iteration within every physical time step. All such iterations are constituted by a fifth order Runge-Kutta calculation, requiring evaluation of the residuum five times. Steps of this evaluation are as follows:

- For each cell, compute velocity gradients.
- For each node, computed viscous fluxes.
- For each node, compute convective fluxes and their derivatives.
- For each edge, compute flows across edge cross sections.
- Compute the residuum.

We observe boundary conditions directly. For sources, we set the velocity and compute the pressure, for sinks, we set the pressure and compute the velocity. At solid or elastic boundaries we implement the no slip condition by zeroing out velocities. We note that it is also possible to implement boundary conditions indirectly, incorporating flow through boundaries into the flux evaluation[8]. This, in theory, could deliver better accuracy.

In order to create the finite volume grid, we implemented algorithmic generation of structured test cases (see Figure 5), and integrated the Gdel[2] library for generic node distributions. The Gdel library allows us to create Delaunay triangulations and tetrahedralizations on the GPU, using CUDA (see Figure 6). We used this to create our unstructured test cases.

We have implemented a Direct3D-based visualisation system to display the computed flow, and a Lua-based script framework to define flow problems and configure the simulation.

6. Preliminary validation

We examined the operation of our finite volume solution without interaction from elastic components. For this, we

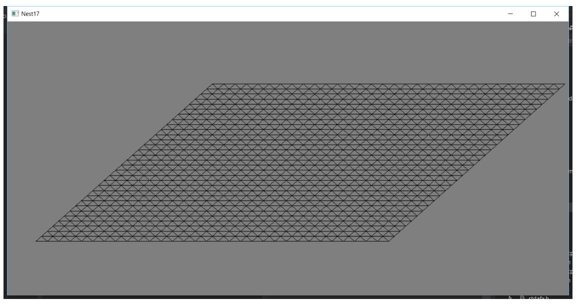

Figure 5: *Regular diamond grid.*

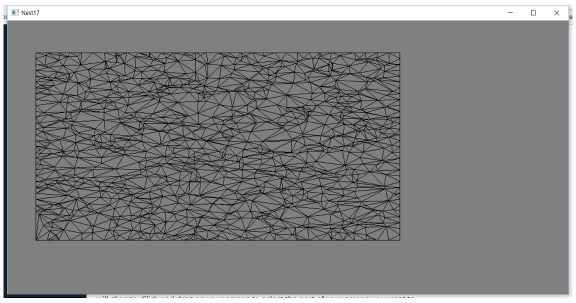

Figure 6: *Unstructured grid created using the Gdel library.*

simulated flow in a tube using both structured and unstructured grids, with an obstacle and without it, with various settings and flow parameters (e.g. Reynolds-number). While this produced no new conclusions, we could verify that the simulation provides the expected results (figure 7) and is able to handle thin membranes as boundaries (figure 8).

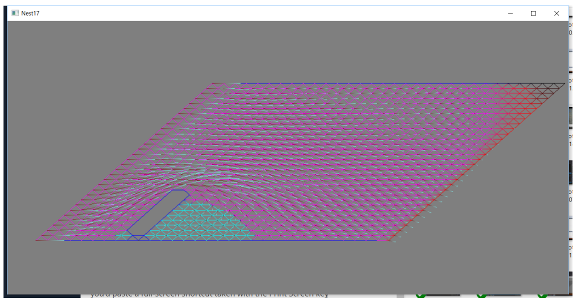

Figure 7: *A vortex forms behind the obstacle in the flow in a tube.*

In future research, we plan to validate our solution against flow scenarios measured on a mock-up system with rubber leaflets and water. The motion of the mock-up can be measured with depth field capturing devices even when occluded by fluid[4] and depth resolution can be improved by filtering[9]. Then, the quantitative comparison of measured and simulated motion would be possible.

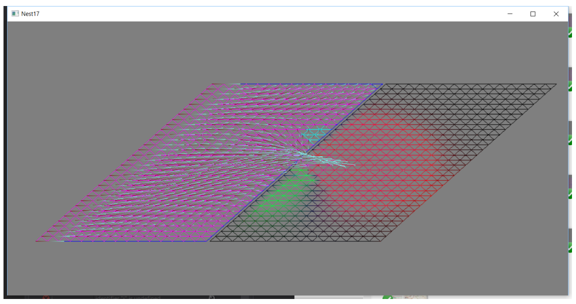

Figure 8: *Flow passing through a small gap between thin membranes.*

7. Future use

The finite volume method can be used in connection with the elastic finite element model[10] to simulate leakage in the closed state of the heart valve. For this, our implementation would need to be able to exchange data with the elastic finite element simulation. This, after sufficient validation, could already solve the targeted medical problem of comparing different strategies for heart valve leaflet alignment.

We opted for the control volume formulation of the finite volume approach because of its ability to work with a moving, adaptive grid suffering large deformations. The above formulas all hold for moving grids, when velocities are interpreted as relative velocities over a moving grid. However, interpolation between the different grids belonging to different time steps has to be implemented. The algorithm for this is known[13]. This would enable the simulation to work with dynamic boundary conditions, allowing a comparison with other methods, and with an SPH implementation[11] in particular. Considering that the finite volume solution has superior theoretical accuracy (along with a higher computational burden because of the need to continuously rebuild or update the deformed grid), it can be used as a validator of the SPH implementation.

Finally, the finite volume computation could be integrated with elastic simulation to create a coupled system. Such a solution would be a significant advancement, depending on the accuracy achieved.

Acknowledgements

This work has been supported by VKSZ-14 SCOPIA and OTKA K-124124.

References

1. HE Abdel Baieth. Physical parameters of blood as a non-newtonian fluid. *International journal of biomedical science: IJBS*, 4(4):323, 2008.

2. Thanh-Tung Cao, Ashwin Nanjappa, Mingcen Gao,

and Tiow-Seng Tan. A GPU accelerated algorithm for 3d delaunay triangulation. In *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 47–54. ACM, 2014.

3. Alexandre Joel Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of computational physics*, 135(2):118–125, 1997.
4. Sundara Tejaswi Digumarti, Gaurav Chaurasia, Aparna Taneja, Roland Siegwart, Amber Thomas, and Paul Beardsley. Underwater 3d capture using a low-cost commercial depth camera. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016.
5. Ansys Fluent. 12.0 Theory Guide. *Ansys Inc*, 5, 2009.
6. R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, dec 1977.
7. CW Hirt, Anthony A Amsden, and JL Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of computational physics*, 14(3):227–253, 1974.
8. Hiroaki Nishikawa. Accuracy-preserving boundary flux quadrature for finite-volume discretization on unstructured grids. *Journal of Computational Physics*, 281:518–555, 2015.
9. László Szirmay-Kalos. Filtering and gradient estimation for distance fields by quadratic regression. *Periodica Polytechnica Electrical Engineering and Computer Science*, 59(4):175–180, 2015.
10. Tamás Umenhoffer, Márton Tóth, László Szécsi, Ágota Kacsó, and Balázs Benyó. Aortic root simulation framework for valve sparing aortic root replacement surgery. In *Proceedings of the Workshop on the Advances of Information Technology, 2018*. BME, 2018.
11. Tamás Umenhoffer, Márton Tóth, László Szécsi, Ágota Kacsó, and Balázs Benyó. Aortic root simulation using smoothed particle hydrodynamics. In *Proceedings of the Workshop on the Advances of Information Technology, 2018*. BME, 2018.
12. Yong Zhao and Ahmed Forhad. A general method for simulation of fluid flows with moving and compliant boundaries on unstructured grids. *Computer methods in applied mechanics and engineering*, 192(39):4439–4466, 2003.
13. Yong Zhao and Baili Zhang. A high-order characteristics upwind FV method for incompressible flow and heat transfer simulation on unstructured grids. *Computer methods in applied mechanics and engineering*, 190(5):733–756, 2000.