Comprehensive Pre- & Post-Processing for Numerical Simulations in Cardiac Surgery Assistance

Nicolai Schoch, Fabian Kißler, Markus Stoll, Sandy Engelhardt,
Raffaele de Simone, Ivo Wolf, Rolf Bendl, and Vincent Heuveline

Preprint No. 2015-02

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

Affiliation of the Authors

Nicolai Schoch[a,1], Fabian Kißler[a], Markus Stoll[b], Sandy Engelhardt[c], Raffaele de Simone[d], Ivo Wolf[c], Rolf Bendl[b], Vincent Heuveline[a]

[a] *Engineering Mathematics and Computing Lab (EMCL), Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Germany*
[b] *Department of Radiation Oncology, University Hospital Heidelberg, Germany*
[c] *Division of Medical and Biological Informatics (MBI), German Cancer Research Cencer (DKFZ), Germany*
[d] *Department of Cardiac Surgery, University Hospital Heidelberg, Germany*
[1] *Corresponding Author: Nicolai Schoch, nicolai.schoch@iwr.uni-heidelberg.de*

Impressum

Heidelberg University
Interdisciplinary Center for Scientific Computing (IWR)
Engineering Mathematics and Computing Lab (EMCL)
Speyerer Str. 6,
69115 Heidelberg
Germany

Comprehensive Pre- & Post-Processing for Numerical Simulations in Cardiac Surgery Assistance

Nicolai Schoch, Fabian Kißler, Markus Stoll, Sandy Engelhardt, Raffaele de Simone, Ivo Wolf, Rolf Bendl, and Vincent Heuveline

August 12, 2015

Abstract

Patient-specific biomechanical simulations of the behavior of soft tissue play an important role in surgery assistance systems. They can provide surgeons with valuable insights and additional information for diagnosis and therapy. In this work, we focus on the surgical treatment of incompetent mitral valves by means of minimally-invasive mitral valve reconstructions (MVR). We aim at supporting MVR surgery by providing adequate FEM-based soft tissue simulations, which simulate the behavior of the patient-individual mitral valve subject to natural forces during the cardiac cycle and to surgical manipulation during MVR.

In order to forster the applicability of such simulations, they need to be fully integrated in often complex biomechanical modeling workflows, which cover the complete pipeline from imaging and segmentation to meshing, model setup and simulation. Like this, an automated execution of all preprocessing steps and of the simulation itself can be enabled. However, patient-specifity and workflow automation often do not fit together, and are sometimes conflicting, such that the respective biomechanical models and the results of corresponding simulation scenarios quickly turn unrealistic or even misleading for surgery assistance.

Therefore, we have set up a comprehensive pipeline of geometry analytics tools and simulation setup operators, which allow for defining and facilitating patient-specific MVR surgery simulation scenarios. Based on these scenarios and on the therein referred pipeline-produced simulation input data, we can feed and run our dedicated MVR simulation application which is built on top of the open-source C++ FEM software toolkit HiFlow^3.

In our work, we thus explain the concept and implementation of the pre- and post-processing operators of such MVR simulations, and point out specific features and problems arising therein. Also, we describe the integration of our operator pipeline into the framework of the Medical Simulation Markup Language (MSML), which shall allow for an automated and more general usage. Finally, we present a set of exemplary results of an application of our operator pipeline to real MVR patient data and explain how the respective model and simulation represent real anatomy and the surgical intervention of an MVR. Concluding, we discuss the suggested setup and give an outlook on possible future developments.

1 Introduction and Motivation

Biomechanical simulations of the human body play an increasingly important role in today's clinical and surgical treatment processes, in training systems, assistance systems, and for clinical diagnosis. Especially when dealing with very complex surgical operations, such as a mitral valve reconstruction, numerical simulations may provide surgeons with valuable insights and additional information for diagnosis and therapy, see, e.g., [16].

This is also part of the vision of the German research project *Cognition-Guided Surgery*, where an intelligent surgery support system recognizes situations in the operation room and suggests adequate surgical actions to the operating surgeon.

In our work, we specifically consider the surgical treatment of incompetent mitral valves by means of minimally-invasive mitral valve reconstructions (MVR). During an MVR, an artificial annuloplasty ring is implanted into the heart in order to reestablish the functionality of an incompetent mitral valve [8]. For a successful operation this requires the surgeon to have a well-established experience and detailed surgical expert knowledge, in order to considerately adapt the course of the annuloplasty ring implantation strategy to respective current situations and intraoperative assessments [9].

Surgery simulations may support surgeons in their work, however, the applicability and usefullness of simulation-based surgery support for MVR is controversially discussed [16], as the simulation technology in comparable assistance systems can not yet fully satisfy the surgical requirements, see, e.g., [3, 10, 12]. On the one hand, surgery assistance by means of simulations has to deliver simulation results that are highly accurate and efficiently computed in almost real-time. On the other hand, these simulations and their underlying models not only need to integrate the above-named surgical expert knowledge with respect to the MVR itself, and hence accordingly adapt the model and simulation boundary condition specifications to current states in the OR. It also has to analyze and consider patient-individual organ geometries and material properties, and accordingly represent these in the biomechanical model setup.

In the previous work [13], which specifically addresses the expert knowledge-based adaptation capability, we suggested a concept for a workflow to overcome this challenge, by integrating a biomechanical simulation for MVR into a knowledge-based surgery assistance system, see Figure 1.

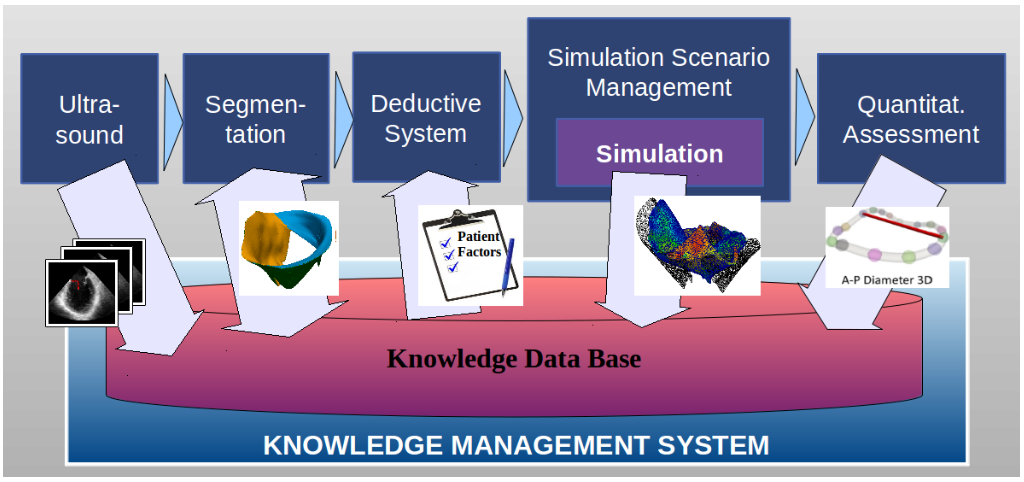


Figure 1: Workflow concept of the knowledge-based simulation-supported MVR surgery assistance system.

Built up on a common data and knowledge base, we semantically represent and perform the various surgery-preparing steps taking place before and during MVR.

Using 4D transesophageal ultrasound images, we carry out semi-automatic segmentations of the valvular apparatus and perform clinically important quantifications, such as of the area of the coaptation zone of the mitral valve and of the bending of the leaflets [4]. In a subsequent step, a deductive system then interprets the given patient-specific data and geometries by means of semantically represented surgical guidelines, in order to perform reasoning and to derive potentially suitable annuloplasty rings for implantation during MVR. Based on this information – the segmented mitral valve geometry and the type of annuloplasty ring which is to be implanted –, the biomechanical soft tissue simulation must accordingly be set up by means of defining the corresponding simulation scenario (i.e., input geometries, boundary conditions, material properties, etc.). This is what this work aims at facilitating in a patient-individual and expert-knowledge-considering way, and what is represented in the workflow draft, Figure 1, as *simulation scenario management*. It then allows for simulating the wanted behavior of the mitral valve subject to the natural external factors of influence (like forces and momentums) and with respect to the virtual implantation of the annuloplasty ring. In the last step of the MVR assistance workflow, the simulation results are presented to a surgeon for a final visual and quantitative assessment, and thus help him during the decision process, where the type of ring and the way of sewing it onto the native annulus of the valve have to be determined.

Based on this work, we want to provide the tools and operators to automatically process the information coming from the imaging step, namely ultrasound imaging and subsequent semi-automatic segmentation, and from the deductive system, which detects potentially suitable rings. We have therefore set up a comprehensive pipeline of miscellaneous Python- and VTK-based scripts. First, it transforms the segmented mitral valve geometries into 3D images and further into 3D volume meshes which are compatible for simulations based on the Finite Element Method (FEM) in general, and particularly for FEM simulations based on our inhouse-developed HiFlow^3 MVR simulation application. Second, it considers the above mentioned specific expert knowledge-based information about the annuloplasty ring implantation strategy by means of processing the respective data into the boundary condition data structures for the simulation.

In the following, we begin with a short introduction to the anatomy and functionality of the human heart and the mitral valve, and point out the biomechanical framework and requirements of our MVR simulation for surgery assistance. Subsequently, we explain the structure and implementation of our preprocessing pipeline, and how it is integrated into the framework of the Medical Simulation Markup Language (MSML) for generalist usage. Using real MVR patient data, we also present an exemplary result of our operator pipeline, and explain how the respective biomechanical model and simulation are set up, which illustrates the applicability of our work in the biomechanical modeling workflow and in clinical environments. Concluding, we discuss the suggested setup and give an outlook on possible future developments.

2 Context and Setup of MVR Simulations

In this section, we first give a short introduction to the anatomy and functionality of the human heart and of the mitral valve, specifically pointing out the circumstances which require surgical treatment by means of a mitral valve reconstruction (MVR).We explain the surgical procedure of an MVR and describe how we want to facilitate the virtual simulation of such a procedure and its physiological results by providing the corresponding boundary conditions for the setup for our numerical simulation.

2.1 Short Excursion: Anatomy of the Human Heart and the Surgical Procedure of an MVR

In order to understand the structure and functionality of the human heart, and especially of the mitral valve, we make a brief excursion into anatomy. The anatomic structure of the mitral valve comprises leaflets, annulus, chordae tendinae, papillary muscles, and underlying left ventricular myocardium, see Figure 2 (a).

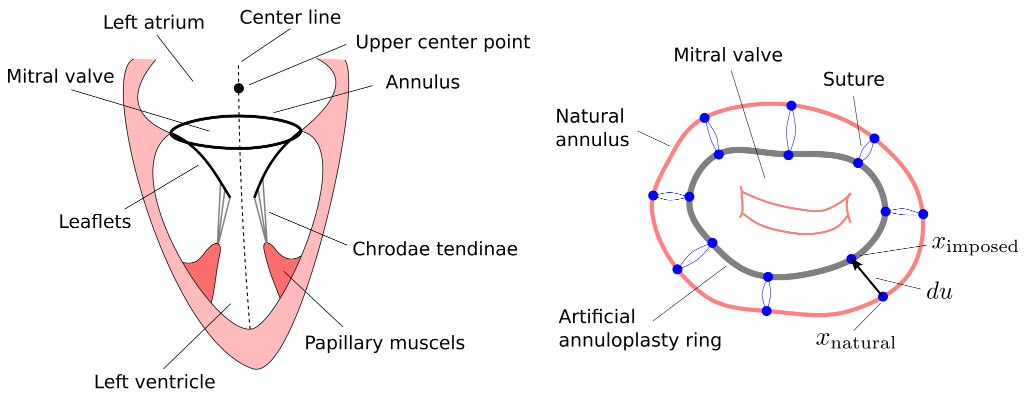


Figure 2: (a) Draft of the left ventricle of the human heart and the mitral valve. (b) Draft of an annuloplasty ring implantation during an MVR.

The *mitral valve*, also known as the *bicuspid* or *left atrioventricular valve*, is a dual-flap valve in the heart, which lies between the left atrium and the left ventricle, and which along with the aortic valve helps control the flow of blood. During *diastole*, a properly-functioning mitral valve opens as a result of increasing pressure from the left atrium as it fills with blood. The open valve thus facilitates the passive flow of blood into the left ventricle. Diastole ends with atrial contraction, and the mitral valve closes to prevent a reversal of blood flow.

The mitral valve has two *leaflets* (the anterolateral leaflet and the posteromedial leaflet) to guard the opening and closing procedures. A ring-shaped fibrous tissue named *annulus* surrounds the opening. The leaflets are prevented from prolapsing into the left atrium by the action of the so-called *chordae tendineae*. These inelastic tendons are attached at one end to the posterior surface of the valve leaflets and at the other to the papillary muscles, which are fingerlike projections from the wall of the left ventricle. When the left ventricle contracts at the beginning of systole and when the intraventricular pressure forces the valve to move up and close, they ensure that the coapting (i.e., touching) leaflets keep together by preventing the valve from over-opening in the wrong direction. The blood is thus prevented from flowing back to the left atrium, see Figure 2 (a).

The size of the *coaptation zone*, i.e., the area where the leaflets are pressed together by the blood during the left ventricle contraction, is important for the valve's closing functionality. Incomplete closure of the valve may result in *mitral insufficiency*, which is a regurgitation or backflow of blood into the atrium. Improperly coapting leaflets and hence leaking valves may be corrected by *mitral valve annuloplasty* or *reconstruction*, a common surgical procedure that aims at restoring proper leaflet coaptation, cf. [8, 9]. During this procedure, a stiff artificial annuloplasty ring prosthesis is sewed and implanted onto the natural annulus in order to stabilize, reshape and shrink the valve geometry, and to hence allow for proper closing after the surgical intervention, see Figure 2 (b) and Figure 3 (a).

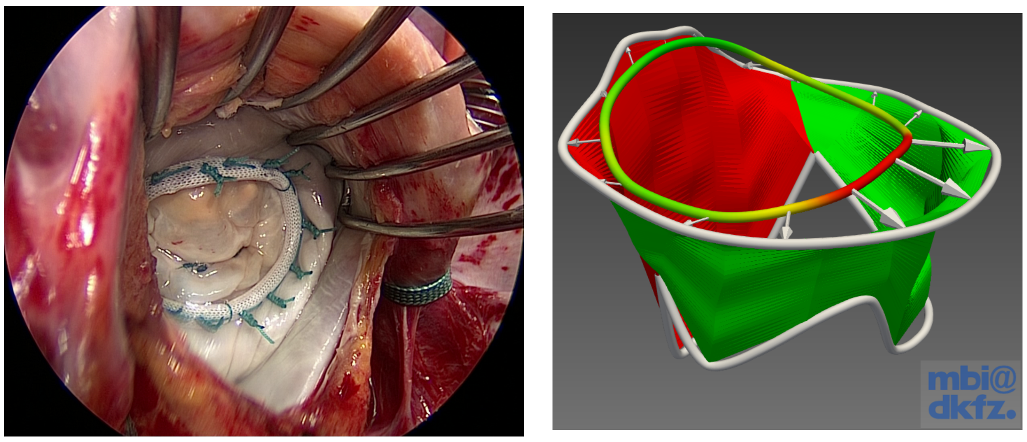


Figure 3: (a) Endoscopic image of the mitral valve captured during an MVR, showing the fixation and implantation of an annuloplasty ring prosthesis. (b) Optimal (distance-based) annuloplasty ring sizing in MITK.

There are different *annuloplasty ring types* which vary in size and shape, in order to allow for perfect fitting to the patient-specific annulus. Surgeons in different institutions have access to different certified and commercially available subsets of rings, so, for instance, in the German Cancer Research Center (DKFZ) and the Department for Cardiology in the University Clinic Heidelberg, Germany, we are provided with the *Carpentier* and *Physio II* ring prostheses only.

During the MVR operation, the surgeon has to decide – on the basis of his experience and his expert knowledge – which ring fits best to the patient-individual annulus, which is crucial for the proper restoring of the valve closure.

The optimal shape can be derived on the basis of a set of *surgical MVR guidelines*, considering patient-individual factors like age, sex, former heart diseases, current diagnostic characteristics and symptoms (fainting, dizziness, swellings, etc.), or specifically with respect to the heart: size of the annulus, curvature, thickness or perforation of the leaflets, length, ruptures and fibroelasticity of the chordae, healthy or degenerative valves, and many more. – These factors are semantically represented in our knowledge base, such that a *deductive system*, in which these surgical expert guidelines are implemented, can perform an intelligent reasoning, and hence provide decision support to the surgeon.

The optimal sizing of rings is identified using the open-source software *Medical Imaging Toolkit* (MITK) [11], and a therein provided plugin named *Mitralyzer* [7], such that it minimally reshapes the natural geometry of the annulus at specifically sensitive points, while still allowing for best coaptation of the leaflets, cf. Figure 3 (b).

We refer to [2, 9], as well as to thorough online documentation[1] for additional background information.

2.2 Setup of the FEM-based MVR Soft Tissue Simulation

In the context of surgery assistance applications, elastomechanical simulations are intended to reproduce the effects of prescribed displacements, forces and momentums on soft tissue, subject either to surgical manipulation through surgery instruments or to natural organ motion. The

[1]Detailed information on the mitral valve: http://www.TheMitralValve.org/.

equations of *elasticity theory* describe the relation between stress and strain in soft tissues, and hence set the basis for an FEM simulation of a body's deformation behavior.

In our work, we simulate the behavior of the mitral valve subject to natural blood pressure and to the artificial implantation of the annuloplasty ring prosthesis, cf. Figure 4.

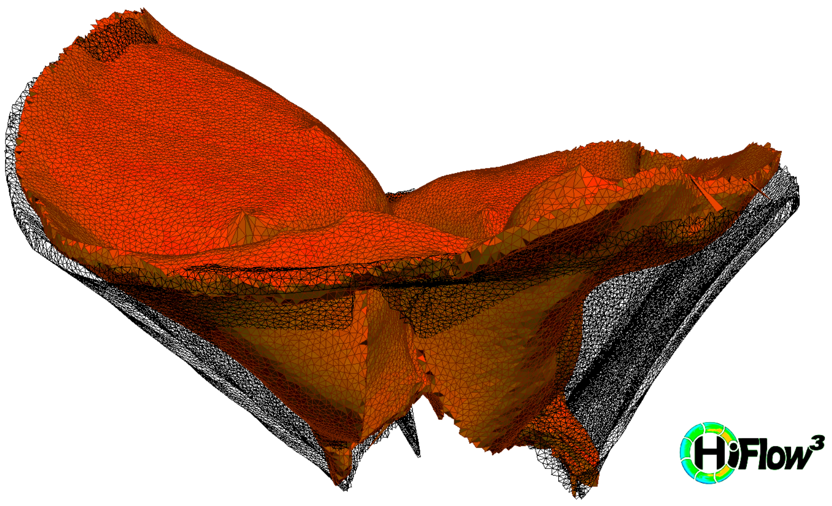


Figure 4: Visualization of HiFlow^3-based MVR simulation results, showing the initial geometry of the mitral valve leaflets before surgery (wireframe), and the manipulated geometry after implantation of an annuloplasty ring prosthesis and subject to blood pressure (colored).

Based on the formulation as a boundary value problem (for background information, see, e.g., [1] for standard elasticity problems, or [17] for elasticity problems considering the occurrence of contact), we can prescribe different types of boundary conditions that shall be taken into account during the simulation.

On the one hand, we need to model the implantation of the annuloplasty ring, and on the other hand, we have to consider the blood pressure onto the leaflets and the back-pulling effects of the chordae. Additionally, the event of contact must be handled appropriately. In order to consider all these constraints in the simulation, we implemented the data structures for four different types of boundary conditions (BCs) and have to accordingly facilitate their setup in our preprocessing pipeline:

- *Pointwise Dirichlet BCs:* We gather all prescribed point displacements arising from the annuloplasty ring implantation step during MVR surgery, where points on the natural annulus are moved to the ring surface by means of a sewing procedure, cf. Figure 2 (b). In a preprocessing step, the corresponding points x_{natural} and displacement vectors $du = x_{\text{imposed}} - x_{\text{natural}}$ have to be computed using the given information on valve and ring geometries, and then processed into the Dirichlet data structures during the simulation setup.
- *Facetwise Neumann BCs:* Modeling the blood pressure onto the valve's leaflets during the cardiac cycle, we implemented facetwise Neumann BCs, which represent a time-dependent force on the boundary surface elements. Time-dependence is given in two ways: On the one hand, we interpolate by trigonometric functions the strength of the blood pressure, according to a mean pressure profile by Wigger, see [2, 5]. On the other hand, we have changing

force directions, e.g., during systole from ventricle-centerline-directed to towards-the-center-of-the-atrium-directed due to the ventricle contraction (such that the valve closes), and similarly, during diastole it is bottom-of-the-valve-directed (such that the valve opens), cf. Figure 2 (a). We therefore need to precompute the respective geometric information (center point of atrium, centerline of ventricle) in our preprocessing pipeline, and mark the leaflets' surface facets accordingly, such that they can be detected and processed by the simulation.

- *Pointwise Neumann BCs:* In order to account for the action of the chordae tendinae, which prevent the leaflets from prolapsing into the atrium during systole, we implement a pointwise Neumann BC, to pull down the leaflets when they are pushed up too far due to the blood pressure (i.e., due to the effects of the above facetwise Neumann BCs). The strength of the pull is increased in dependence of the distance between the chordae's attachment points on the leaflets and a representation of the initial location of the top point of the papillary muscles (which is assumed to move along with the ventricle over the course of systole), cf. Figure 2 (a). We have to represent the (primary) chordae's attachment points on the leaflets according to a mean distribution as suggested by [10], and account for the pulling forces by means of a bottom-of-the-ventricle directed set of vectors. These vectors hence coincide with and point into the direction of the chordae.
- *Facetwise Penalty Contact BCs:* Accounting for contact in an instationary simulation first of all requires the search for potential contact occurrences in every time step. This is a highly critical and very costly task, and the algorithms for detecting contact are of utmost importance. The number of operations needed to check $O(N)$ surfaces of a discretized domain for potential contact is $O(N^2)$, such that carrying out these checks in every time step easily lets the search dominate the overall time for the computation. The contact analysis hence needs to be designed to be most efficient, and complexity must be maximally reduced. – We therefore edit the boundary mesh in a cheap preprocessing step in a way such that, firstly, the two leaflets are assigned with two different IDs, and secondly, the upper and lower sides of the leaflets can be distinguished, cf. Figure 7. Like this, the contact search directly requires only approximately $\frac{1}{16}$ of the original computation time. For more information on the thus facilitated contact search algorithm, we refer to [14]. Once the contact facets are found, we apply a penalty method to penalize the advent of contact.

In the following section, we explain, how this setup of data structures, which is suitable for FEM soft tissue simulations, is computed by means of a dedicated pipeline of preprocessing operators.

3 Setup of Data Structures and Implementation Issues

In order to have a clearly defined basis and namespace for the subsequent description, we introduce some notation: Let S' be the segmented mitral valve geometry which is a polygonal grid, whose 2D cells are triangles. In our case, we obtain this segmentation from the *German Cancer Research Center* (DKFZ) and in the *Department of Cardiology* of the *University Hospital in Heidelberg*, Germany. For detailed information on the image and geometry acquisition process, starting with 3D+t transesophageal ultrasound imaging and ending with 3D segmentations of the mitral valve leaflets, we refer to specific documentation: For their inhouse-developed open-source *Medical Imaging Toolkit*[2] (MITK), see [11], for the dedicated *Mitralyzer* Plugin, see [7], and we point out the work of Engelhardt et al. [4] with respect to the specific semi-automatic segmentation steps taken to obtain the geometries, which we use for further preprocessing.

[2]Official Website of the *Medical Imaging Toolkit* (MITK): http://www.mitk.org/.

In a previous work, we presented the *Medical Simulation Markup Language*[3] (MSML), which allows for simplifying the biomechanical modeling workflow, see [15]. In the following, we start our workflow pipeline using a sequence of general image data processing operators, which we already integrated into the MSML for usage in the context of universal preprocessing pipelines, also beyond MVR-related tasks. Namely, these operators are: a VTK[4] -based `SurfaceToVoxelData` operator (which converts a 2D polydata surface in 3D space in a `vtp` file into a 3D voxel image in a `vti` file), a VTK-based `MarchingCube` operator (which extracts a polygonal surface mesh in a `vtu` file of an isosurface from voxel data in the `vti` file, and allows for smoothing, too), and a CGAL[5] -based `SurfaceToVolumeMesh` operator (which generates a 3D volume mesh in a `vtu` file inside the above mentioned polygonal surface mesh).

Altogether, these operators allow for obtaining from the segmented mitral valve geometry S' a volume mesh V of the mitral valve which is suitable for FEM-based simulations. The mesh V is an unstructured grid, whose 3D cells are tetrahedra.

The surface S of V can be extracted as a polygonal grid with the VTK classes `vtkDataSet-SurfaceFilter` or `vtkGeometryFilter`. The geometry filter is more flexible than the surface filter, but for our purposes, both filters suffice.

Since S is a closed surface, we can compute its unit outer normal vector field N with the VTK class `vtkPolyDataNormals`. The corresponding normal vector of a cell c of S is denoted by $N(c)$. With the same filter, we can also compute a normal vector field N' for the surface S'. Since S' is not closed, we need to make a choice concerning the orientation of N'. In the following, as shown in Figure 5a, N' is oriented towards the center line of the left ventricle (see again Figure 2 (a)).

Finally, the artificial annuloplasty ring, which tightens the natural annulus, is represented as a polygonal grid and will be denoted by R.

Based on this notation, we now start outlining the series of operators, which we conceptually developed and implemented in order to build up a comprehensive operator pipeline that allows for the automated setup of patient-individual, situation-adaptive and surgical expert knowledge-based simulation scenarios for MVR operations. It is designed to produce patient-individual simulation scenario setups through processing patient-specific data (ultrasound images and medical records) from our clinic partners, and it is situation-adaptive and also considers surgical expert knowledge as it employs and integrates into its MVR ring implantation scenarios the results of an MVR guidelines-based deductive system. This system derives patient-individual possible suitable sets of annuloplasty rings on the basis of a surgery knowledge data base and specifically important patient factors, as already mentioned above in section 2.1.

First, we need to explain how, on the basis of the segmentation, we derive the upper and lower sides of the leaflet geometry and accordingly annotate the data structure, see 3.1. Second, we present an algorithm to distinguish the anterior and posterior leaflets in order to facilitate an efficient contact simulation, see 3.2. In section 3.3, we describe the MVR-specific setup of pointwise Dirichlet boundary conditions considering the annuloplasty ring implantation procedure. Last, in section 3.4, we depict the patient-individual representation of the back-pull effects of the chordae tendinae on respective mitral valve geometries in terms of pointwise Neumann boundary conditions.

[3]Online sources of the *Medical Simulation Markup Language* (MSML): see the GitHub-based source code repository on https://github.com/CognitionGuidedSurgery/msml, and the corresponding documentation on http://msml.readthedocs.org/.

[4]Official Website of the *Visualization Toolkit* (VTK): http://www.vtk.org/.

[5]Official Website of the *Computational Geometry Algorithms Library* (CGAL): http://www.cgal.org/.

Following these preprocessing steps, we execute the HiFlow^3-based MVR elasticity simulation in order to obtain simulation results according to the given MVR scenarios. In the post-processing procedure, i.e., after the HiFlow^3-based MVR elasticity simulation, we conclude with a stress visualization step, explained in section 3.5. We fully integrated the pre- and postprocessing operators into the framework of the MSML, which is described in the last section 3.6 of this chapter.

In our work, we make use of algorithms and functionalities available in the open-source libraries and toolkits VTK and specifically Python-VTK, as well as the fundamental packages for scientific computing with Python, NumPy[6] and SciPy[7]. We hence assume that future users of our algorithms have these packages installed and linked on their workstations. A documentation for the VTK classes and methods used in the following sections can be found on the internet[8], too.

3.1 Derivation of upper and lower side of the leaflets (Preprocessing)

We describe two methods for finding the leaflet's surface facets on the upper and lower side of the mitral valve. Thereby, the upper side of the valve's surface is the side on which the blood pushes during diastole. Accordingly, we define the surface's lower side. We approached this problem in two ways, each using the normal vector fields N or N'.

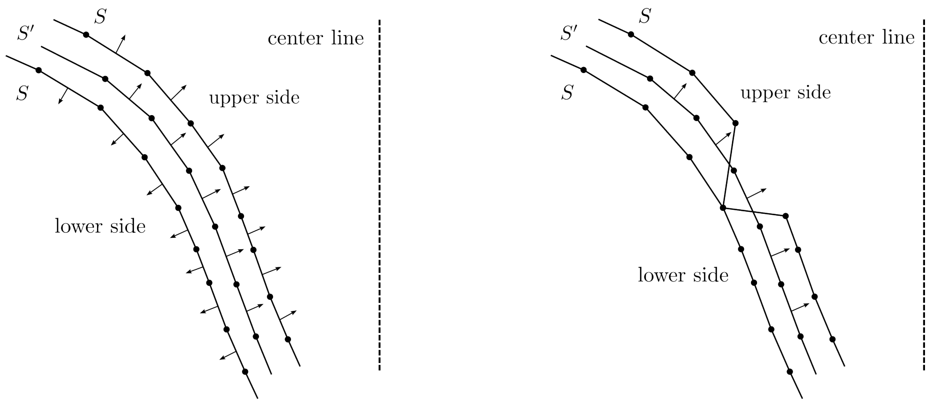
The first idea is based on a local comparison between both vector fields, see Algorithm 1: Let c be a cell in S and let v be one of its vertices. With the VTK method `FindClosestPoint()` which is a member of the class `vtkCellLocator` we find a closest point to v in S' and a cell c' which contains this point. Now, c is on the upper side of the mitral valve's surface if the normals $N(c)$ and $N'(c')$ roughly point in the same direction, i.e., the Euclidean inner product of both vectors is greater than zero. Figure 5a shows a cross-section of the surfaces S and S' and their normal fields characterizing the upper and lower side of S.

Algorithm 1 Upper and lower side (via comparison of cell normals)

```
1:  for all cells $c$ in $S$ do
2:      choose a vertex $v$ of $c$
3:      find a closest point $p'$ to $v$ in $S'$
4:      find a cell in $S'$ that contains $p'$
5:      if $N(c) \cdot N'(c') > 0$ then
6:          $c$ is on the upper side
7:      else
8:          $c$ is on the lower side
9:      end if
10: end for
```

The disadvantage of this method is that it completely relies on a correct computation of both normal vector fields. We experienced that the filter `vtkPolyDataNormals` produces correct

[6]Official Website of *NumPy*: http://www.numpy.org/.
[7]Official Website of *SciPy*: http://www.scipy.org/.
[8]Official Website of *VTK* including documentation: http://www.vtk.org/documentation/.



(a) Ideal world case. (b) Exception case.

Figure 5: Cross-sections of the segmented mitral valve leaflet geometry and the surface of the volume mesh with normal vector fields.

results for the segmented mitral valve geometry, whereas the normals of the surface of the volume mesh were not always computed correctly, since the results were not deterministic.

Consequently, we developed another algorithm, see Algorithm 2, that solves the same problem, but does not rely on the normals of the surface mesh anymore: As before, choose a cell c in S. We introduce a variable called *upLowIndicator* which, as the name suggests, will indicate whether the cell c is classified as a cell on the upper or lower side of the surface. We proceed as follows.

First, set *upLowIndicator* to 0. Then, we iterate over the vertices v of c. Using the same VTK methods as in Algorithm 1, we find a closest point p' to v in S' and a cell c' that contains p', and consider the plane with origin p' and normal $N'(c')$, which can be described as the set of points x that satisfy the plane equation $N'(c') \cdot (x - p') = 0$. This plane separates $\mathbb{R}^3$ into two half spaces, a positive and a negative half space. If v is contained in the positive half space, i.e., $N'(c') \cdot (v - p') > 0$, we increase the variable *upLowIndicator* by one. Otherwise we decrease the indicator variable by one. The evaluation of the term $N'(c') \cdot (v - p')$ can be done with the VTK method `Evaluate()` of the class `vtkPlane`.

After the iteration the indicator variable is either positive or negative. It cannot be equal to zero, since the cells of the surface mesh are triangles which have an odd number of vertices. In the positive case, we classify the cell c as a cell on the upper side, otherwise it will be classified as a cell on the lower side.

The reason why we still need to iterate over all the vertices of a cell is as follows: In an ideal world, since the segmented mitral valve geometry S' would be nicely nested inside the volume mesh, and be completely covered by its surface S as it is the case in the Figure 5a, so we would not need to iterate over all vertices. However, locally the surface S can have sharp bends, e.g., in cases where the volume meshing operator is too coarse, as it is shown in Figure 5b, such that a cell can have vertices in positive and negative half spaces. Hence, by iterating over all the vertices of a cell we obtain a more reliable classification, with only few relatively small locally limited exceptions.

Figure 6 shows the color-coded representation of the up/low cell identification of a surface mesh that was generated with Algorithm 2.

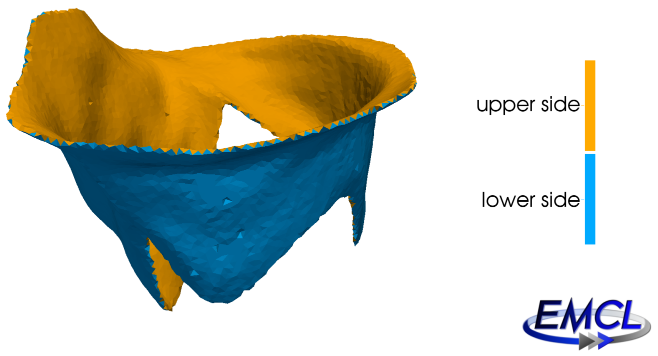


Figure 6: Color-coded up/low cell identification of a surface mesh using Algorithm 2.

Algorithm 2 Upper and lower side (via half spaces)

```
 1: for all cells $c$ in $S$ do
 2:   $upLowIndicator \leftarrow 0$
 3:   for all vertices $v$ of $c$ do
 4:     find a closest point $p'$ to $v$ in $S'$
 5:     find a cell in $S'$ that contains $p'$
 6:     if $N'(c') \cdot (v - p') > 0$ then
 7:       $upLowIndicator \leftarrow upLowIndicator + 1$
 8:     else
 9:       $upLowIndicator \leftarrow upLowIndicator - 1$
10:     end if
11:   end for
12:   if $upLowIndicator > 0$ then
13:     $c$ is on the upper side
14:   else
15:     $c$ is on the lower side
16:   end if
17: end for
```

3.2 Derivation of anterior or posterior leaflets (Preprocessing)

In this section, we depict a method to determine, whether a cell on the surface mesh belongs to the anterior (anterolateral) or posterior (posteromedial) leaflet of the mitral valve.

The segmented mitral valve geometry, which comes from the DKFZ and the Department of Cardiology of the University Hospital in Heidelberg, is annotated with vertex IDs. These MITK-produced IDs indicate, whether the vertex belongs to the anterior or posterior leaflet, or whether it is a sewing point of the surgical suture which fixes the artificial annuloplasty ring to the natural annulus. In the following algorithm, see Algorithm 3, we use the IDs of the segmented geometry for a classification of the cells of the surface mesh.

Algorithm 3 Anterior or posterior leaflet

```
1: for all cells $c$ in $S$ do
2:   $antPostIndicator \leftarrow 0$
3:   for all vertices $v$ of $c$ do
4:     $N \leftarrow 1$
5:     $pointNotClassified \leftarrow true$
6:     while $pointNotClassified$ do
7:       find closest $N$ vertices to $v$ in $S'$
8:       for all closest points $p'$ do
9:         if ID of $p'$ is 'anterior' then
10:          $antPostIndicator \leftarrow antPostIndicator + 1$
11:          $pointNotClassified \leftarrow false$
12:        end if
13:        if ID of $p'$ is 'posterior' then
14:          $antPostIndicator \leftarrow antPostIndicator - 1$
15:          $pointNotClassified \leftarrow false$
16:        end if
17:      end for
18:      $N \leftarrow N + 1$
19:    end while
20:  end for
21:  if $antPostIndicator > 0$ then
22:    mark $c$ as anterior
23:  else
24:    mark $c$ as posterior
25:  end if
26: end for
```

Let c be a cell in S. Similar to the problem of the classification of cells on the upper and lower side of the mitral valve, we will iterate over the vertices of c and decide for each vertex to which leaflet it belongs. Then, we mark the cell accordingly, based on the classification of its vertices. For this purpose, as in Algorithm 2, we introduce an indicator variable *antPostIndicator* and initialize it with the value zero. Now, we classify the vertices v of the cell. An intuitive approach is the following: For any vertex v we find a closest point p' in S' and mark v according to the ID of p'. Indeed, this would work for most points in S, but it can happen that the closest point we find in S' is marked as a sewing point for the artificial ring and thus carries no information about the leaflets. Consequently, we need to find more closest points to v.

This can be done very efficiently using the VTK method `FindClosestNPoints()`, which is a member of `vtkKdTreePointLocator`. Here N is the number of closest points we want to find in S' to v. Starting with just one point, we increase N until we find a point that is not a sewing point. If v is closest to a point on the anterior leaflet we increase the indicator variable *antPostIndicator* by one, otherwise we decrease by one. After the iteration over the vertices we can mark the cell. It is on the anterior leaflet if the indicator variable is greater than zero, otherwise it is on the posterior leaflet.

With Algorithm 3, one can of course also classify certain subsets of cells on the surface like, for example, cells that had been marked with an indicator according to Algorithm 2. Figure 7 shows a corresponding color-coded representation of the anterior/posterior cell identification of the upper side of a surface mesh that was generated with Algorithm 3.

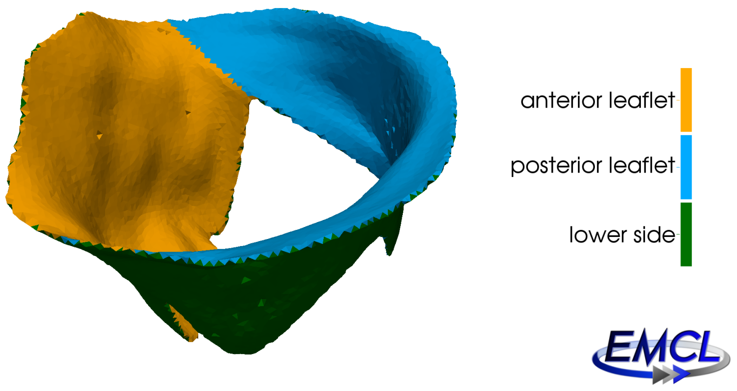


Figure 7: Color-coded anterior/posterior cell identification of the upper side of a surface mesh, generated with Algorithm 3.

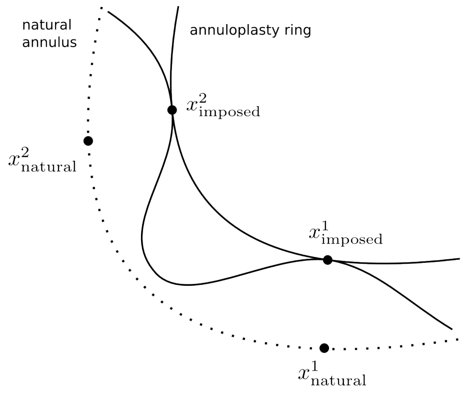
3.3 Derivation of the Pointwise Dirichlet BCs (Preprocessing)

During MVR, the natural annulus is sutured to a stiff artificial annuloplasty ring at selected points, see sections 2.1 and 2.2. The points on the annulus and the annuloplasty ring are denoted by x_{natural} and x_{imposed}, respectively. Figure 2 (b) shows a draft of the annulus, the artificial ring and the suture. During this surgical procedure, the specific points on the annulus are pulled towards the ring.

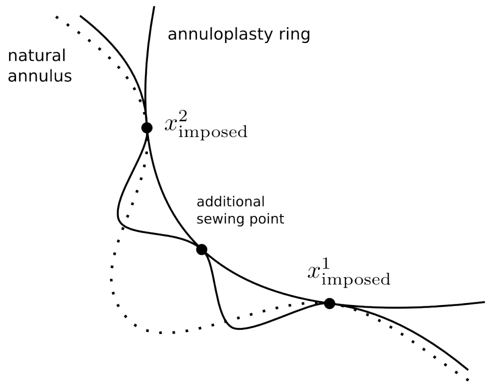
As shown in Figure 8a, it commonly occurs that folds are formed between the sewing points. If a fold is too large, it is recommended to suture additional points to the ring, in order to correct the unintended deformation, which would enormously stiffen the tissue in the subsequent simulation, see Figure 8b.

This procedure is modeled by imposing Dirichlet boundary conditions on a finite set of prescribed suture points on the natural annulus. These points are represented by accordingly marked vertices of S', given in consecutive order in counter-clock-wise direction around the annulus, where each of them was assigned a unique ID, see Table 1.

For each suture point $\tilde{x}^i_{\text{natural}}$ in S', there is a corresponding point x^i_{imposed} on the annuloplasty ring R, which is tagged with the same ID. Given these pairs of distinguished points, for every $\tilde{x}^i_{\text{natural}}$, we compute a closest point x^i_{natural} on the mitral valve's surface S and the resulting displacement vector $du^i = x^i_{\text{imposed}} - x^i_{\text{natural}}$, see Figure 9a.



(a) Fold between two points.



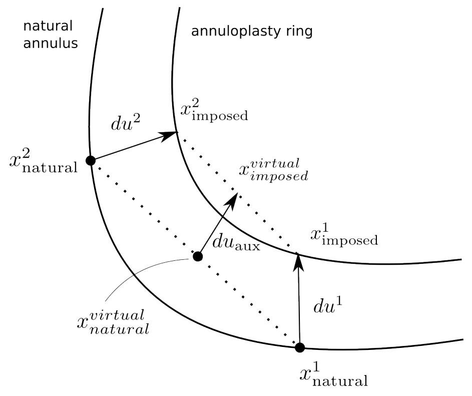
(b) Situation after correction.

Figure 8: Draft of the natural annulus and the annuloplasty ring with suture points. The dotted line indicates the position of the natural annulus before the application of the corrections.

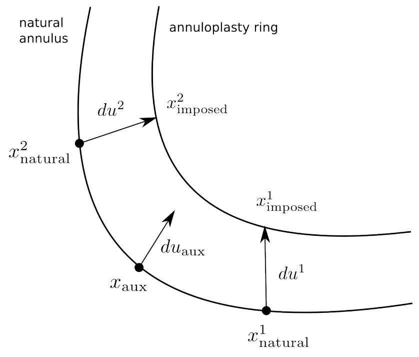
Table 1: List of annulus point IDs and their corresponding anatomical names and meanings[a] (set around the annulus in clockwise direction).

0	Anterolateral commissure	8	Posteromedial commissure
1 - 3	Control point	9 - 11	Control point
4	Saddle-horn	12	Posterior annulus
5 - 7	Control point	13 - 15	Control point

[a] List of annulus point IDs as chosen and prescribed by our clinic partners in the DKFZ.



(a) Computation of additional boundary conditions.



(b) Situation after closest point search.

Figure 9: Draft of the natural annulus and the annuloplasty ring with suture points and corresponding displacement vectors.

We also take the unintended formation of folds between the prescribed suture points into account: As shown in Figure 9a, for every pair of consecutive suture points on the surface S and on the ring R, we compute the midpoints of the line segments connecting them. We call those midpoints "virtual points", $x_{\text{natural}}^{\text{virtual}}$ and $x_{\text{imposed}}^{\text{virtual}}$, since, in general, they are neither elements of S nor of R. For the additional suture point x_{aux}, we use a closest point to $x_{\text{natural}}^{\text{virtual}}$ in S with the corresponding displacement vector $du_{\text{aux}} = x_{\text{imposed}}^{\text{virtual}} - x_{\text{natural}}^{\text{virtual}}$, as shown in Figure 9b. The closest point search can be carried out with the VTK method `FindClosestPoint()` of the class `vtkKdTreePointLocator`. Algorithm 4 summarizes the above explained steps.

Algorithm 4 Pointwise Dirichlet BCs

```
1: Prescribed points and displacements:
2: for all suture points $\tilde{x}_{\text{natural}}^i$ in $S'$ do
3:     find the corresponding point $x_{\text{imposed}}^i$ on $R$
4:     find a closest point $x_{\text{natural}}^i$ on $S$ to $\tilde{x}_{\text{natural}}^i$
5:     compute the displacement vector $du^i = x_{\text{imposed}}^i - x_{\text{natural}}^i$
6: end for
7: Auxiliary points:
8: for all pairs of consecutive distinguished points on $S$ and $R$, $(x_{\text{natural}}^i, x_{\text{natural}}^{i+1})$ and
   $(x_{\text{imposed}}^i, x_{\text{imposed}}^{i+1})$ do
9:     compute the midpoint $x_{\text{natural}}^{\text{virtual}}$ of the line segment connecting $x_{\text{natural}}^i$ and $x_{\text{natural}}^{i+1}$
10:    compute the midpoint $x_{\text{imposed}}^{\text{virtual}}$ of the line segment connecting $x_{\text{imposed}}^i$ and $x_{\text{imposed}}^{i+1}$
11:    compute the displacement vector $du_{\text{aux}} = x_{\text{imposed}}^{\text{virtual}} - x_{\text{natural}}^{\text{virtual}}$
12: end for
```

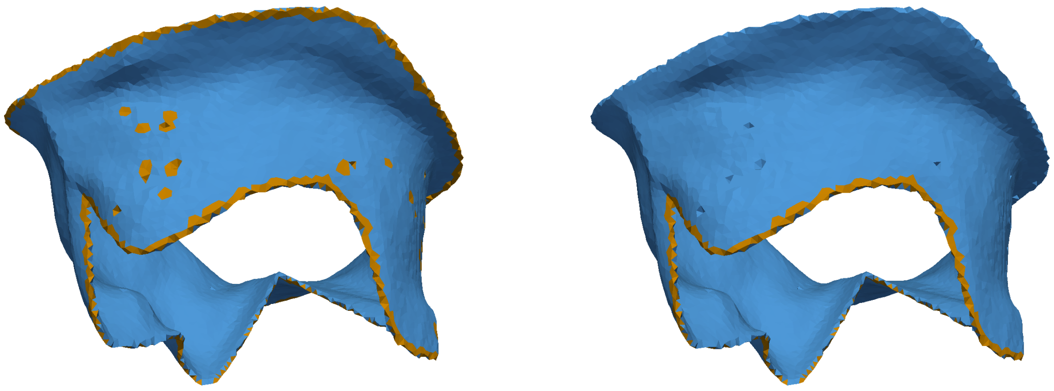
3.4 Derivation of the Pointwise Neumann BCs (Preprocessing)

As described in sections 2.1 and 2.2, we account for the action of the chordae tendinae on the leaflets by imposing pointwise Neumann BCs. Here, we present a method to distribute the chordae's attachment points on the lower rim of the leaflets by means of a *Farthest-First Traversal* algorithm, see [6]. In the following, we define the lower rim of the leaflets and describe how it is computed as a subset of cells of S. Afterwards, we depict the *Farthest-First Traversal* algorithm.

The lower rim of the leaflets can be computed as follows: First, we find the cells on the upper and lower side of the cardiac valve's surface with algorithm 2 in section 3.1. Figure 6 shows a color-coded representation of this classification of cells. Let C' be the set of cells on the lower side of the surface neighboring a cell on the upper side. Generally, the set of neighbors to a given cell can be found with the VTK method `GetCellNeighbors()` from the class `vtkPolyData`, where one cell is defined as a neighbor of another cell if it shares a vertex. The set C' is depicted in Figure 10a.

Second, find a cell in C' such that one of its vertices has a minimal z-coordinate among all the vertices of cells in C'. The lower rim C of the cardiac valve's surface is the connected component of cells in C' that contains the previously specified cell with minimal z-coordinate, see Figure 10b.

The connected component can be computed in the following way: Initialize C as the set that contains the cell with the minimal z-coordinate. Then, recursively add cells to C which are in C' and neighbor a cell in C.



(a) Upper and lower rim of the leaflets with irregularities.

(b) Lower rim of the leaflets.

Figure 10: Color-coded representation of the cells on the lower side of the mitral valve's surface neighboring a cell on the upper side.

Having computed the lower rim of the mitral valve's surface, we can now use the *Farthest-First Traversal* algorithm to distribute the chordae's attachment points on the vertices of the lower rim. The *Farthest-First Traversal* algorithm, see Algorithm 5, is a greedy algorithm which computes a 2-approximation to the optimal solution of the k-center clustering problem. This problem reads as follows:

Given a metric space X, a finite subset $S \subseteq X$ and a positive integer k, find a subset $T \subseteq S$, the center set, with $\#T = k$, which is a solution to the problem

$$\min_{T \subseteq S,\ |T|=k} \max_{p \in S} d(p, T).$$

Algorithm 5 Farthest-First Traversal

```
1: Choose $x \in X$.
2: $T \leftarrow \{x\}$
3: while $\#T < k$ do
4:     Find $x \in \arg\max_{p \in S} d(p, T)$
5:     $T \leftarrow T \cup \{x\}$
6: end while
```

We use a greedy algorithm, since there exist no methods that solve this optimization problem while still running in polynomial time. Figure 11 depicts a distribution of 16 points on the lower rim of the mitral valve's surface, computed as explained in this subsection, with the hence combined Algorithm 6.

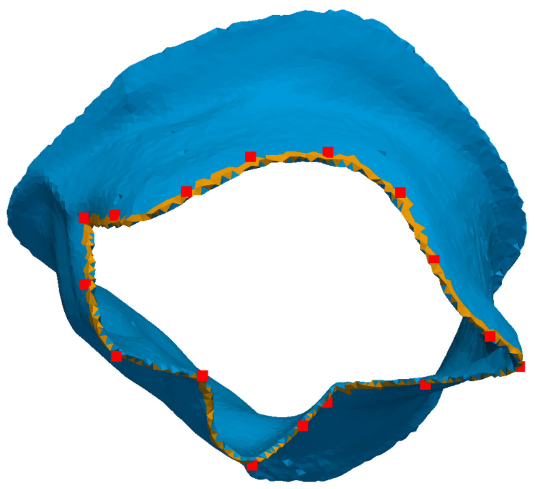


Figure 11: Distribution of 16 points on the lower rim of the mitral valve's surface computed with the Farthest-First Traversal algorithm.

Algorithm 6 Pointwise Neumann BCs

```
1: find cells on the upper and lower side of $S$, see algorithm 2
2: find the set $C'$ of cells on the lower side of $S$ neighboring a cell on the upper side
3: find a cell in $C'$ such that one of its vertices has a minimal $z$-coordinate among all the
   vertices of cells in $C'$
4: the lower rim $C$ of $S$ is the connected component of cells in $C'$ that contains the
   previously specified cell
5: perform the Farthest-First Traversal algorithm, see algorithm 5, on the set of vertices
   of cells in $C$ to distribute the chordae's attachment points
```

After executing these preprocessing steps and having produced the corresponding output data, our HiFlow^3-based MVR elasticity simulation (see [14]) is fully defined and set up, such that we can run the simulation to obtain mitral valve behavior simulation results according to the given MVR scenarios, cf. Figure 14.

In the subsequent post-processing procedure, i.e., after the MVR elasticity simulation, we conclude with a stress analysis and visualization step, in order to provide the surgeons with additional information on the *von Mises stress* distribution and behavior of the mitral valve leaflets after the virtual annuloplasty ring implantation. Critically high stress values – occurring after the implantation of a suboptimal annuloplasty ring – may indicate the yielding of the tissue, or in worse cases even lead to stenosis, ring dehiscence, or obstruction of the ventricular outflow tract.

3.5 Von Mises stress evaluation and visualization (Postprocessing)

In this section, we outline the *von Mises stress* measure, which is commonly used as an indicator for material failure, and depict its computation using VTK methods. A detailed derivation of the formula for the *von Mises stress* would entail a portrayal of hydrostatic and deviatoric components of stress and strain tensors, Hooke's Law, and strain energy density, which is not the aim of this work. Here, we only define terms that are necessary to understand the definition of the *von Mises stress*, and refer to detailed online documentation[9] .

Let ϵ and σ denote the infinitesimal strain tensor and the Cauchy stress tensor. By Hooke's Law, stress can be written as a function of strain and vice versa:

$$\sigma = \lambda \operatorname{tr}(\epsilon) \, \mathrm{I} + 2\mu\epsilon, \quad \epsilon = \frac{1}{2\mu}\sigma - \frac{\lambda}{2\mu(3\lambda + 2\mu)} \operatorname{tr}(\sigma) \, \mathrm{I},$$

where λ and μ are Lamé's parameters. Both tensors, strain and stress, can be decomposed into a hydrostatic and deviatoric part, where the hydrostatic part is related to the change in volume and the deviatoric part is related to the change in shape. We define

$$\sigma_{\text{hyd}} = \frac{1}{3} \operatorname{tr}(\sigma) \, \mathrm{I}, \quad \sigma_{\text{dev}} = \sigma - \sigma_{\text{hyd}} \quad \text{and} \quad \epsilon_{\text{hyd}} = \frac{1}{3} \operatorname{tr}(\epsilon) \, \mathrm{I}, \quad \epsilon_{\text{dev}} = \epsilon - \epsilon_{\text{hyd}}.$$

From Hooke's Law, one can derive that deviatoric stress is proportional to deviatoric strain:

$$\sigma_{\text{dev}} = 2\mu\epsilon_{\text{dev}}.$$

The scalar value of the *von Mises stress* is defined in terms of deviatoric stress:

$$\sigma_{\text{vM}} = \sqrt{\frac{3}{2} \; \sigma_{\text{dev}} : \sigma_{\text{dev}}} = \sqrt{\frac{3}{2} \; (\sigma_{\text{dev}})_{ij} \, (\sigma_{\text{dev}})_{ij}}.$$

With the above-mentioned equations, the term σ_{dev} can also be interpreted as a function of the deviatoric strain. We remark that there exist other alternative algebraic equation forms for the computation of the von Mises stress, which are investigated on. Here, we refer to standard literature for material sciences and to the above mentioned the webpage on continuum mechanics, only, and go on with the computation of the von Mises stress in the implementation of our postprocessing operator.

The simulation produces three scalar arrays u_0, u_1 and u_2, which contain the x-, y-, and z-coordinates of the displacement vectors u for every vertex in the volume mesh V. In order to visualize the deformed state of the mitral valve, we use the VTK class `vtkWarpScalar` to successively add the scalars in the arrays u_i to the coordinates of the respective points in V, and hence obtain the vertex coordinates of the deformed object.

The computation of the von Mises stress entails three steps: First, we use the VTK class `vtkArrayCalculator` to compute an array of displacement solution vectors for every vertex in V from the scalar arrays u_0, u_1 and u_2. Then, one can compute the strain tensor with the class `vtkCellDerivatives` using the previously generated array of displacement vectors. Finally, again using the class `vtkArrayCalculator`, we compute the von Mises stress from the strain tensor according to the above equation.

Figure 12 shows the results of an exemplary MVR simulation scenario at a stage, where the valve is half-closed, including the von Mises stress distribution. The corresponding scenario was defined in the context of the collaborative research center *Cognition-Guided Surgery* (SFB TRR 125) in a cooperation with the cardiac surgeons at the University Hospital in Heidelberg.

[9] See comprehensive documentation on the website *ContinuumMechanics*: http://www.continuummechanics.org.

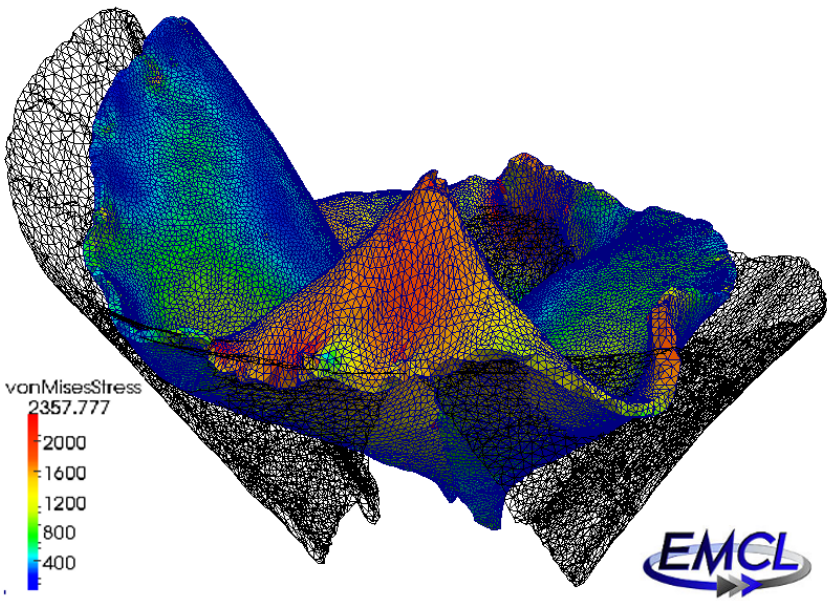


Figure 12: MVR simulation scenario result, obtained with HiFlow^3, showing the mitral valve leaflets before (wireframe) and after (colored) the implantation of an annuloplasty ring prosthesis. The von Mises stress distribution (in MPa) as computed in our postprocessing operator is projected and color-coded onto the leaflet surface.

Since we not only plan to provide surgeons with the pure simulation of the deformation behavior of the mitral valve, but also want to visualize the stress distribution in every time step, we need to apply the von Mises stress evaluation script to the current result of every simulation time step, too.

We therefore make use of the Python module `glob`, which basically returns an iterator of files in a directory using shell pattern matching. We specify the simulation output directory (where our HiFlow^3-based MVR simulation application stores the simulation results in `vtu` or `pvtu` format) in order to have the von Mises stress evaluation script run over all simulation results one after the other. Using `iglob` even allows to perform this task in real-time as soon as new simulation results are available, as the data is not all stored in one buffer or list in memory, but can be read out one at a time.

3.6 Integration of the operators into the MSML and facilitation of an automated biomechanical modeling and simulation pipeline for MVR

In a previous publication [15], we presented the *Medical Simulation Markup Language* (MSML), and how it simplifies the biomechanical modeling workflow.

In short, the MSML modeling approach aims at constructing patient-specific biomechanical models (which are suitable for FEM-based simulations) from tomographic data. It seeks to flexibly describe both the preprocessing workflow leading to the model and the simulation with the model itself in a generalized way, therein integrating a range of different tools that are used within the workflow (from segmentation, to meshing, and model setup), cf. Figure 13.

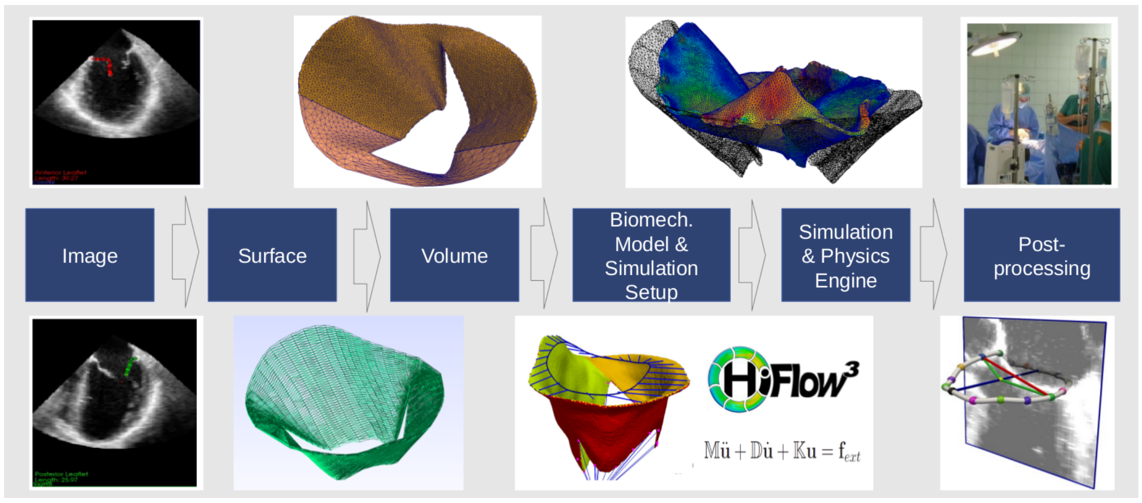


Image
Surface
Volume
Biomech. Model & Simulation Setup
Simulation & Physics Engine
Post-processing
HiFlow³
$M\ddot{u} + D\dot{u} + Ku = f_{ext}$

Figure 13: MSML workflow for biomechanical MVR modeling and simulation.

The foundation of the MSML is the `MSMLalphabet`, which allows to define and specify a comprehensive range of parameters (e.g. the object's material regions, physics model equations, solid mechanics or fluid dynamics setups, boundary conditions, etc.) as well as operators (e.g. mesh size calculators, etc.). Subsets of the parameters may then be employed in the `MSMLscene` and in the `MSMLworkflow` to describe the respective object under consideration, the simulation scenario (including boundary conditions, FE solvers, etc.), and the way how the model and the simulation shall be set up in a preprocessing workflow, or how the simulation results shall be processed in a postprocessing workflow. All components of the alphabet are defined by means of a flexibly extensible hierarchy of XML files.

The different items in the workflow are associated with certain data types and linked by so-called *operators*.After completion of all preprocessing steps, so-called *export operators* or *exporters* take over the whole set of information that was computed so far, and process it into physics engine-specific simulation scenario descriptions, which can then be used for executing the simulation by means of those specific physics engines (e.g. SOFA, Abaqus, FEbio, HiFlow^3, etc.).

Being implemented as a hybrid Python/C++ system, the MSML scheme allows for all high level functionalities (such as XML-parsing, compatibility checking, and operator calls) to be performed by the Python part, and for all operators to be executed either as function calls or as command line tools. Like this, the flexible inclusion of additional external tools (such as algorithms based on VTK, CGAL, etc.) can easily be guaranteed.

Finally, this setup allows for defining both the `MSMLscene` and the `MSMLworkflow` in MSML XML files, which can be read and processed by the MSML in order to automatically perform all steps declared, from image to simulation. We refer to the open-source GitHub code repository[10] of the MSML and to the example showcases for more information.

Concluding, the central idea of the MSML is hence, to not only describe the final biomechanical simulation, but also the workflow how the biomechanical model is constructed from tomographic data. This is exactly what we intend with the integration of our MVR operators and what we describe in the following.

In the current preliminary setup, we took up the above depicted MVR simulation preprocessing

[10]Online sources of the *Medical Simulation Markup Language* (MSML): see the GitHub-based source code repository on https://github.com/CognitionGuidedSurgery/msml.

and postprocessing operators into the MSML framework, and integrated them for usage with the HiFlow^3 simulation exporter, in order to allow for the automated setup and execution of HiFlow^3-based MVR simulation scenarios.

Therefore, we introduced the operators to the MSML alphabet, such that they can be referred to by the MSML, and we defined the physical and logical data types of their inputs/outputs, such that these can be processed in MSML pipelines. All operators-deduced data are transferred into MSML-internal data structures (such as geometry information, mesh properties, material IDs, boundary condition specifications, etc.), and hence available for further usage and processing by the respective physics engine exporters.

In our current implementation, we built up a special `mitral` exporter, which inherits its general structure and features from the `HiFlow`^{3} exporter, and includes additional MVR simulation-specific information, in order to set up specific HiFlow^3-based MVR simulation scenarios. For instance, it accounts for patient-specific mitral valve geometries, for the implantation of the annuloplasty ring prosthesis via Dirichlet BCs, for the integration of blood pressure profiles via Neumann BCs, and for upcoming contact between the two leaflets during the cardiac cycle via Contact BCs.

Currently, these pieces of information are retrieved by the MVR operators and processed via MSML data structures in order to be handed over to the specific `mitral` exporter for the creation of HiFlow^3 MVR simulation scene files. We implemented the corresponding interface between the MSML (and the MSML-based data structures, respectively) and the HiFlow^3-based MVR simulation. Finally, we showed its compatibility by running MSML-produced simulation setups, cf. Figure 12. This requires the simulation code to, e.g., facilitate time-dependent boundary conditions, or to integrate geometry information into pressure directions, which cannot yet be handled by other physics engines such as SOFA or Abaqus.

However, we intend to generalize this setup in some future work, such that not only HiFlow^3-based MVR simulations can be set up and executed, but in a way such that other exporters can be fed and executed, too. Of course, first of all, this requires those other simulation engines to be capable of processing the information correctly, which at this stage is not yet the case. We remark that this is automatically taken care of in the MSML, since, due to the fact that every exporter holds a tree of so-called *compatible nodes*, consistency between the MSML description and the physics engine is ensured via compatibility checking.

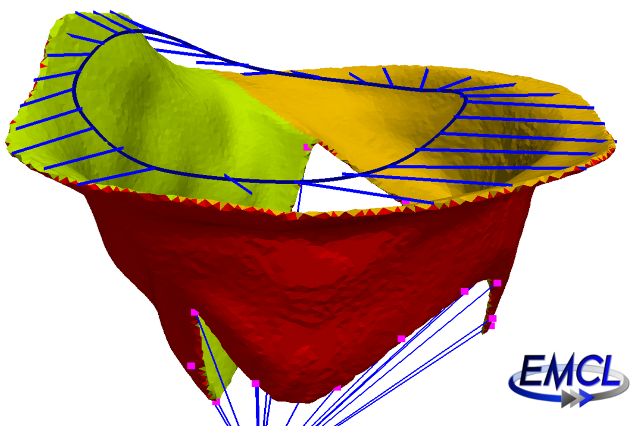


Figure 14: Visualization of an example MVR simulation scenario setup, which is derived by the presented series of MVR preprocessing operators in the MSML.

In total, the described setup and implementation allows for using the MSML as a middleware between all steps and tools in the modeling pipeline, see Figure 13. By means of the integration of our MVR operators (and of their respective inputs/outputs) into the MSML, we enable the execution of a fully-automated, holistic setup of patient-individual MVR simulation scenarios. Furthermore, as opposed to just simply writing a single script to combine the operators in a fixed workflow, we allow anyone to use our developments via a generalistic and easily accessible interface, with respect to which one does not need to care about inputs/outputs, file formats, shared operator and pipeline activities, and compatibility of data structures anymore. This is of great value not only for surgeons and clinical staff due to easy and automated usage in surgery assistance systems, but also for developers, as flexibility is maintained throughout the complete workflow and complex tools that have once been integrated can be accessed and used via this general interface.

Figure 14 visualizes an example for an outcome of the MSML-based MVR preprocessing pipeline, representing the complete set of boundary conditions and geometry features of our biomechanical MVR model. The underlying geometry data were produced in both HiFlow^3-readable format and in VTK-based file formats, such that the combined set is compatible with the software *MITK* and the special *CGS Workbench* release version, so our clinic partners can directly use and visualize it in the context of their surgery assistance environments and devices. Like this, we guarantee for maximal transparency and understandability of our model and simulation setups for the operating surgeons. The corresponding features for the MSML workflow integration in MITK are implemented as part of the *MSML CLI developments*.

Finally, we conclude with the sample setup of an MSML XML file, which can be used for processing ultrasound-imaging-based segmentation data of the mitral valve and optimally placed annuloplasty ring prostheses into patient-individual MVR simulation scenarios, cf. Listing 1.

Listing 1: MSML description for MVR preprocessing/modeling/simulation.

```
<msml:msml xmlns:msml="http://sfb125.de/msml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://sfb125.de/msml">

   <variables>
      <var name="input_mv_surf_mesh"
          value="PatientID_XXX/Segmentations/MitralValve_annotated_ts0.vtp"
          logical="TriangularSurface" physical="vtp" role="input" />
      <var name="input_mv_ring"
          value="PatientID_XXX/AnnuloplastyRings/OptimallyPlaced/PhysioII_size34.vtp"
          logical="Object" physical="vtp" role="input" />
   </variables>

   <scene>
      <object id="mitralvalve">
         <mesh>
            <linearTet id="mvVolMesh"
                     meshId="\${anHf3matIDsProducer.inpMeshFile}"/>
         </mesh>
         <material>
            <region id="mvMaterial"
                  indices="\${aVtuToHf3inpIncMVmatIDsProducer.tet_material}">
               <materialID matID="10"/>
               <linearElasticMaterial youngModulus="2350" poissonRatio="0.488"/>
               <mass name="stdMass" massDensity="1.000"/>
            </region>
         </material>
         <constraints>
            <constraint name="bodyConstraint" forStep="\${MVR_simulation}">
               <mvGeometryConstraint mvGeometry="\${anMvGeometryAnalyzer}"/>
               <displacedPointsConstraint points="\${aBCdataForMVRsimProducer.points}"
                        displacements="\${aBCdataForMVRsimProducer.displacements}"/>
               <surfacePressureConstraint indices="\${anHf3matIDsProducer.fixedSet}"
                        pressure="default_cardiac_pressure"/>
               <pointwiseForceConstraint points="\${aBCdataForMVRsimExtender.points}"
                        forces="\${aBCdataForMVRsimExtender.forces}"/>
            </constraint>
         </constraints>
      </object>
   </scene>

   <workflow>
      <SurfaceToVoxelDataOperator id="aSurfaceToVoxelsOperator"
                           surfaceMesh="\${input_mv_surf_mesh}"
                           targetImageFilename="mvImage.vti"
                           accuracy_level="7" smoothing="1"/>
      <vtkMarchingCube id="aVoxelToContourOperator" image="\${aSurfaceToVoxelsOperator}"
                          outFilename="mvVolumeSurface.vtk" isoValue="90"/>
      <CGALMeshVolumeFromSurface id="aVolumeMesher"
                          meshFilename="mvVolumeMesh3D_CGAL.vtk"
                          surfaceMesh="\${aVoxelToContourOperator}"
                          preserveFeatures="false" facetAngle="20"
                          facetSize="0.5" facetDistance="0.4" cellSize="0.5"/>
      <vtuToHf3inpIncMVmatIDsProducer id="anHf3matIDsProducer"
                          volume="\${aVolumeMesher}"
                          surface="\${input_mv_surf_mesh}"/>
      <mvGeometryAnalytics id="anMvGeometryAnalyzer"
                          surface="\${aVolumeMesher}" ring="\${input_mv_ring}"/>
      <mvrBCdataProducer id="aBCdataForMVRsimProducer"
                 volumeMesh="\${aVolumeMesher}" surfaceMesh="\${input_mv_surf_mesh}"
                 ring="\${input_mv_ring}"/>
      <mvrBCdataExtender id="aBCdataForMVRsimExtender"
                 volumeMesh="\${aVolumeMesher}" surfaceMesh="\${input_mv_surf_mesh}"/>
   </workflow>

   <environment>
      <solver linearSolver="CG" preconditioner="SGAUSS_SEIDEL"
            hf3_chanceOfContactBoolean="1"
```

```
            processingUnit="CPU" numParallelProcessesOnCPU="16"
            timeIntegration="Newmark"
            dampingRayleighRatioMass="0.5" dampingRayleighRatioStiffness="0.5"/>
      <simulation>
         <step name="MVR_simulation" dt="0.05" iterations="300"
              visualizeSimResultsEvery1inXtimesteps="5" />
      </simulation>
   </environment>
</msml:msml>
```

4 Summary, Conclusion and Outlook

In our work, we presented a comprehensive set of pre- and post-processing operators for biomechanical simulation scenarios of Mitral Valve Reconstruction operations in surgery assistance. The operators are integrated into the Medical Simulation Markup Language (MSML) and thus allow for generalistic usage in the complex biomechanical modeling workflow.

We depicted the conceptual development and algorithmic implementation of an operator pipeline: from ultrasound-imaging-derived segmentations, via FE mesh generation and specification, to MVR simulation scenarios. It has been shown how this pipeline allows for the automated setup of patient-individual, situation-adaptive and surgical expert knowledge-based simulation scenarios for MVR operation support. On the one hand, the pipeline produces patient-individual simulation scenario setups through processing patient-specific data (ultrasound images and medical parameters). On the other hand, as it employs and integrates into its MVR ring implantation scenarios the results of an MVR guidelines-based deductive system, it is situation-adaptive and also considers surgical expert knowledge. The resulting simulation scenario setups are specifically compatible for subsequent usage with our HiFlow^3-based MVR simulation application. Finalizing the biomechanical modeling workflow, the von Mises stress analytics postprocessing operator can be applied on the simulation results and hence allow for additional surgery support through visualization of critical stress distributions on the valve leaflets caused by surgical manipulation.

Through our developments, we expect to forster the applicability of biomechanical MVR simulations in simulation-supported surgery assistance systems, since we not only allow for the consideration of patient-specific parameters and surgical expert knowledge, but also fully integrated the depicted analytics and setup operators into the biomechanical modeling and simulation workflow framework of the MSML.

This enables a smooth execution of patient-individual, situation-adaptive and expert knowledge-based FEM soft tissue simulations in order to obtain valuable insights and additional information for cardiac diagnosis and therapy. We have shown the applicability by performing sample MVR simulation scenarios including von Mises stress visualization postprocessing on the basis of MSML-produced scenario setups.

Future work may include a more generalized integration of our operators and of their resulting data structure outputs within the MSML framework, e.g., in order to allow for a more generic use with various simulation engines. Also, the model itself can of course be further specified and optimized, considering, e.g., secondary and tertiary chordae, or flow profiles in the ventricle, which have strong influence on the pressure distributions on the leaflet surfaces during the cardiac cycle and hence, on the closing properties of the valve.

5 Acknowledgements

This work was carried out with the support of the German Research Foundation (DFG) within the project I03 of the Collaborative Research Center SFB/TRR 125 'Cognition-Guided Surgery', with the support of projects A01, C02 (MSML framework and implementation), and B01 (mitral valve model and simulation requirements).

References

[1] D. Braess. Finite elemente. *Springer, Berlin Heidelberg*, 2007.

[2] A. Carpentier and et al. Reconstructive surgery of mitral valve incompetence: ten-year appraisal. *Journal of Thoracic and Cardiovascular Surgery*, 79:338–348, 1980.

[3] A. Choi and et al. A novel finite element-based patient-specific mitral valve repair: virtual ring annuloplasty. *Journal of Bio-Medical Materials and Engineering*, 24:341–347, 2014.

[4] S. Engelhardt and et al. Towards automatic assessment of the mitral valve coaptation zone from 4d ultrasound. 2015.

[5] L. Formaggia, A. Quarteroni, and A. Veneziani. Cardiovascular mathematics. – modeling and simulation of the circulatory system. 2009.

[6] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Journal of Theoretical Computer Science*, 38:293–306, 1985.

[7] B. Graser and et al. Computer assisted analysis of annuloplasty rings. *Bildverarbeitung fuer die Medizin*, pages 75–80, 2013.

[8] J. Jokinen and et al. Mitral valve replacement versus repair: Propensity-adjusted survival and quality-of-life analysis. *Journal of Thoracic Surgery*, 84:451–458, 2007.

[9] F. Maisano and et al. Mitral annuloplasty. *Oxford Journals Multimedia Manual of Cardio-Thoracic Surgery*, 0918, 2009.

[10] T. Mansi and et al. An integrated framework for finite-element modeling of mitral valve biomechanics from medical images. *Journal of Medical Image Analysis*, 16:1330–1346, 2012.

[11] M. Nolden and et al. The medical imaging interaction toolkit: challenges and advances. *International Journal of Computer Assisted Radiology and Surgery*, 8:607–620, 2013.

[12] A. Pouch and et al. Semi-automated mitral valve morphometry and computational stress analysis using 3d ultrasound. *Journal of Biomechanics*, 45:903–907, 2012.

[13] N. Schoch and et al. Integration of a biomechanical simulation for mitral valve reconstruction into a knowledge-based surgery assistance system. 2015.

[14] N. Schoch and V. Heuveline. High-performance computing based soft tissue simulation for cardiac surgery assistance. *EMCL Preprint Series*, 04, 2015.

[15] S. Suwelack and et al. The medical simulation markup language – simplifying the biomechanical modeling workflow. *Journal on Studies in Health Technology and Informatics*, 196:394–400, 2014.

[16] E. Votta and et al. Toward patient-specific simulations of cardiac valves: State-of-the-art and future directions. *Journal of Biomechanics*, 46:217–228, 2013.

[17] P. Wriggers. Computational contact mechanics. 2nd edition. *Springer, Berlin Heidelberg*, 2006.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2015-01	Teresa Beck, Martin Baumann, Leonhard Scheck, Vincent Heuveline, Sarah Jones: Comparison of mesh-adaptation criteria for an idealized tropical cyclone problem
No. 2014-02	Christoph Paulus, Stefan Suwelack, Nicolai Schoch, Stefanie Speidel, Rüdiger Dillmann, Vincent Heuveline: Simulation of Complex Cuts in Soft Tissue with the Extended Finite Element Method (X-FEM)
No. 2014-01	Martin Wlotzka, Vincent Heuveline: A parallel solution scheme for multiphysics evolution problems using OpenPALM
No. 2013-02	Nicolai Schoch, Stefan Suwelack, Rüediger Dillmann, Vincent Heuveline: Simulation of Surgical Cutting in Soft Tissue using the Extended Finite Element Method (X-FEM)
No. 2013-01	Martin Schindewolf, Björn Rocker, Wolfgang Karl, Vincent Heuveline: Evaluation of two Formulations of the Conjugate Gradients Method with Transactional Memory
No. 2012-07	Andreas Helfrich-Schkarbanenko, Vincent Heuveline, Roman Reiner, Sebastian Ritterbusch: Bandwidth-Efficient Parallel Visualization for Mobile Devices
No. 2012-06	Thomas Henn, Vincent Heuveline, Mathias J. Krause, Sebastian Ritterbusch: Aortic Coarctation simulation based on the Lattice Boltzmann method: benchmark results
No. 2012-05	Vincent Heuveline, Eva Ketelaer, Staffan Ronnas, Mareike Schmidtobreick, Martin Wlotzka: Scalability Study of HiFlow^3 based on a Fluid Flow Channel Benchmark
No. 2012-04	Hartwig Anzt, Armen Beglarian, Suren Chilingaryan, Andrew Ferrone, Vincent Heuveline, Andreas Kopmann: A unified Energy Footprint for Simulation Software
No. 2012-03	Vincent Heuveline, Chandramowli Subramanian: The Coffee-table Book of Pseudospectra
No. 2012-02	Dominik P.J. Barz, Hendryk Bockelmann, Vincent Heuveline: Electrokinetic optimization of a micromixer for lab-on-chip applications
No. 2012-01	Sven Janko, Björn Rocker, Martin Schindewolf, Vincent Heuveline, Wolfgang Karl: Software Transactional Memory, OpenMP and Pthread implementations of the Conjugate Gradients Method - a Preliminary Evaluation
No. 2011-17	Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Piotr Luszczek: GPU-Accelerated Asynchronous Error Correction for Mixed Precision Iterative Refinement
No. 2011-16	Vincent Heuveline, Sebastian Ritterbusch, Staffan Ronnås: Augmented Reality for Urban Simulation Visualization
No. 2011-15	Hartwig Anzt, Jack Dongarra, Mark Gates, Stanimire Tomov: Block-asynchronous multigrid smoothers for GPU-accelerated systems
No. 2011-14	Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Stanimire Tomov: A Block-Asynchronous Relaxation Method for Graphics Processing Units

$\frac{\partial u}{\partial t} = \nu \Delta u$
$+ (u \cdot \nabla) u + \frac{1}{\rho} \nabla p$
$\frac{\partial u}{\partial t}$