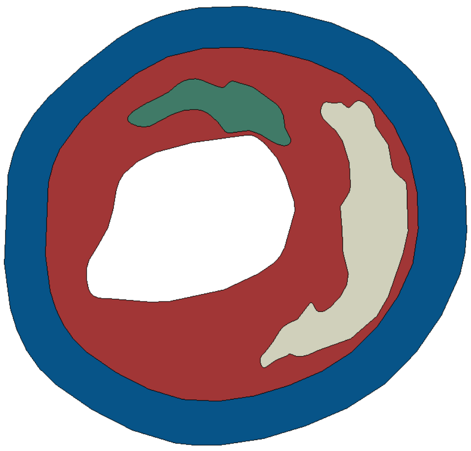
A Novel Approach to Estimating the Material Properties of Atherosclerotic Plaque Tissue

by

R.D. van den Berg

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday, July 18, 2019 at 11:00 AM.



Student number:	4615646	
Project duration:	Februari, 2018 – June, 2019	
Thesis committee:	Prof. dr. A. A. Zadpoor,	TU Delft, supervisor
	Dr. N. Tolou,	TU Delft
	Dr. M. Mirzaali Mazandarani,	TU Delft
	Dr. A. C. Akyildiz,	Erasmus MC, external supervisor

Preface

Before you lies my thesis report: ”A Novel Approach to Estimating the Material Properties of Atherosclerotic Plaque Tissue”. This thesis was written to obtain the degree of Master of Science at the Delft University of Technology.

For this report I chose a slightly different approach from just writing a regular thesis report. I chose to write this report in the format of a journal publication. This means that some of the first steps that I took in the validation process are not described in this report. I did however include some Appendices at the end containing information that would normally not be included in a journal publication. These appendices give a more detailed explanation of certain procedures for the interested reader. The first few Appendices contain information that would have been included in a regular master thesis. Following these first Appendices are the Appendices that are less relevant, for example one containing the MATLAB scripts created for the method I developed.

I did my project at the Erasmus Medical Center (MC) in Rotterdam, where I first did an internship from September 2017 to January 2018 on a slightly different topic. After that, I wrote my literature review at the Erasmus MC on the mechanical characterization of atherosclerotic plaque tissue. For my final graduation project, I then developed a new method for the mechanical characterization of atherosclerotic plaque tissue that I personally believe is better than the current techniques available.

I would like to thank my supervisors at the Erasmus MC Dr. Ali C. Akyildiz and Dr. Frank J.H. Gijsen, for always finding time to answer my questions and supporting me whenever I needed it. I spent hours together with Ali discussing my project and finding solutions for problems I was facing in the project and I am very grateful for all the things I learned during these sessions. Also, I would like to thank Prof. Dr. Stéphane Avril for the skype sessions, and for helping out especially at the beginning of the project when the whole concept of the Virtual Fields Method was completely new to me. Finally, I would like to thank my TU Delft supervisors Prof. dr. A. A. Zadpoor and Dr. J. Zhou for their guidance and kind words.

I hope you enjoy your reading.

R.D. van den Berg
June 2019

Abstract

The majority of cardiovascular clinical events, which are the main causes of mortality and morbidity worldwide, are caused by atherosclerotic plaque rupture. This biomechanical event occurs when the local plaque stresses exceed its strength. The plaque stresses can be assessed by computational models to predict these events. Current approaches to obtaining the plaque material stiffness properties that these models require as input have large computational costs and are therefore far from being implemented for clinical use. This study aims to develop, validate, and apply for the first time, an approach to obtaining the material stiffness properties of atherosclerotic plaque tissue much faster by employing the virtual fields method (VFM). With this method, the virtual work principle is employed with boundary problem specific, kinematically admissible virtual fields to solve energy balance equations for the material stiffness parameters that are of interest. In this study a method is presented for obtaining the virtual fields for the specific application of intraluminally pressurised atherosclerotic plaque tissue. For the purpose of validation, full field displacement maps were computed at 100 mmHg using Finite Element (FE) models based on histological slides of atherosclerotic plaque tissue. To mimic a realistic situation, the resolution and noise levels of a clinical and high frequency ultrasound scanner were used. Although higher resolution deformation maps with smaller noise levels were shown to provide more accurate results, the VFM-based technique demonstrated good performance for both the high frequency and clinical ultrasound scanner settings tested. VFM was also used in a single case study to estimate the c_1 material parameter for a Neo-Hookean incompressible material model in the case of an atherosclerotic human coronary artery. The estimated c_1-values for this case were: 21.5 kPa for diseased intima, 13.3 kPa for lipid, and 23.6 kPa for wall tissue. These values were in good agreement with the reported values from literature. In this study, VFM was applied successfully for the material characterization of atherosclerotic plaques for the first time. It is more attractive than current approaches as it is computationally less expensive and has a great potential to be extended for material characterization of even more plaque components than employed in the current study.

Keywords: Atheroslcerosis, Virtual Fields Method, Material properties, Mechanical characterization, Ultrasound

1. Introduction

Cardiovascular diseases are the main causes of mortality and morbidity worldwide [1]. The majority of cardiovascular clinical events, such as ischemic heart disease and stroke, are mainly triggered by atherosclerosis. Atherosclerosis is a chronic, inflammatory disease of the arterial system [2]. A more detailed description of atherosclerosis can be found in Appendix A. The local manifestations of the disease are referred to as plaques. Some plaques have the flow-limiting nature, either due to stenosis because of their extensive intraluminal progression or due to the thrombogenic events that the plaques lead to if they mechanically fail (plaque rupture). The resulting limited flow might lead to necrosis of the tissues and cells distal to the plaque site, and thus to clinical events.

The currently employed decision-making strategies for preventive surgical interventions are far from optimal as they consider only a single plaque-related criterion, namely the degree of the stenosis [3]. As such, they miss the not-heavily stenotic plaques with high risk of rupture [4]. Hence, there is a clinical need for means of plaque rupture risk assessment. In this regard, biomechanical models hold a great potential.

From a mechanical point of view, rupture is a material failure of plaque tissue. As such, plaque structural stress analyses have the great potential to serve the purpose of clinical rupture risk assessment. To assess plaque stresses, computational models such as finite element (FE) models have been used [5, 6, 7, 8, 9]. The results of atherosclerotic plaque FE models greatly depend on the material behavior of the components of the heterogeneous plaque tissue, especially on the one of the diseased intima (DI) component [7].

Various studies have investigated the material behavior of the atherosclerotic plaque experimentally, using mechanical tests such as uniaxial or biaxial tensile tests and unconfined compression tests [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32] (Appendix B). The majority considered the plaque tissue as a homogeneous structure and extracted the composite, gross tissue properties. The reported stiffness values were shown to span a wide range, even with 2-3 orders of magnitude difference [33]. This great variation was primarily attributed to compositional differences in heterogeneous plaque tissue. Moreover, it has been advocated that more realistic and accurate results would be obtained with testing techniques that mimic the physiological loading environment of the plaque tissue more closely.

For atherosclerotic arteries, inflation tests mimic the physiological loading very well. If combined with an appropriate structural imaging technique, they have the potential to provide the heterogeneous material properties of the atherosclerotic plaques through assessing the properties of the mechanically relevant plaque components. A challenge with this approach is that the material properties cannot be assessed straightforward by applying the standard testing techniques such as tension or compression tests.

A method capable of estimating the material parameters for individual plaque components from inflation tests is inverse FE modelling [34], also referred to as the FE modelling updating (FEMU) method [35]. In this method, a forward FE model for inflation testing is generated and run multiple times. At each run, the material properties of the plaque components in the FE model are changed and the objective function, which measures the difference between the computed deformation and deformation measured in the inflation tests, is reevaluated. By minimizing the objective function, the material properties of the plaque components are estimated. In a recent study, this technique was successfully used by Akyildiz et al. for the estimation of the plaque components from inflation experiments on porcine iliac arteries [34]. The authors also discussed the great potential of the approach for clinical use. However, a major drawback of the approach is that it is computationally expensive as each iteration of the minimization procedure requires running a FE simulation. The procedure may take hours or days, depending on the complexities of the overall minimization problem, and the FE model and the number of the plaque components whose mechanical properties are to be estimated. Hence, the use of the inverse FE for clinical purposes is currently unlikely.

A potentially alternative technique for the material characterization of atherosclerotic plaques from inflation tests or clinical deformation measurements is the virtual fields method (VFM). VFM employs the principle of virtual work, where the energy balance equation (external work = internal stored energy) is used to obtain the material parameters of an assumed constitutive relation [36]. By defining virtual displacement fields proper for the examined boundary value problem, a set of equations are obtained from the virtual work equation. The obtained equations are then used to define an optimization problem to estimate the values of the material properties in the constitutive relation. VFM is a fast approach as it only needs to solve a set of analytical equations at each iteration of the optimization procedure.

VFM has been used for various structural engineering problems [37, 38]. It has also recently been applied to a cardiovascular biomechanical problem to estimate the material stiffness properties of healthy arterial tissue for murine aortas, using the thin wall assumption for the arterial wall [39, 40]. However, this approach cannot be adapted to the case of atherosclerotic plaques as the heterogeneity and the geometry of the plaque tissue make the assumption of homogeneity in the radial direction and the thin-wall-structure assumptions invalid.

The aim of this study is to develop a VFM-based approach for the material characterization of heterogeneous atherosclerotic plaque tissue from ex-vivo inflation test data or in-vivo plaque deformation measurements. The report structure is as follows: First, background information on VFM is provided. Then, the VFM-based approach developed for the particular application of intraluminally pressurized atherosclerotic arteries is introduced. Subsequently, a validation study is done using thought computational experiments, in which the sensitivity of the obtained material properties to noise is evaluated. Finally, a case study for real experimental data is performed.

2. Methods

2.1. Background of VFM

VFM is a method used for the estimation of material properties of a structure. The method is based on the principle of virtual work and requires knowledge of the full field deformation data of the structure of interest and the loading exerted on the structure. For a body in static equilibrium, free of body forces, the principle of virtual work can be described in a three-dimensional Cartesian coordinate system of x, y, and z in the following form [36]:

$$-\int_v (\sigma_{ij}\frac{\partial \delta u_i}{\partial x_j})dv + \int_s (t_i \delta u_i)ds = 0, \quad i = x, y, z, \quad j = x, y, z \tag{1}$$

The first term in this balance equation is the internal virtual work (IVW), which is the integral of the double contraction of the Cauchy stress tensor (σ_{ij}) with the virtual strain field ($\frac{\partial \delta u_i}{\partial x_j}$), over the entire volume (v) of the body in the deformed configuration. The second term in the equation describes the external virtual work (EVW), which is given as the integral of the dot product of the traction (t_i) in the deformed configuration with the virtual displacement field (δu_i) over the surface (s) that the traction is applied on. The Cauchy stress tensor (σ_{ij}) is a function of the material properties and the deformation of the structure. If the applied traction and the full field deformation data are available, by defining virtual displacement fields (δu_i), which are kinematically admissible, the balance equation can be rewritten to solve for the material coefficients of the constitutive relation that is assumed for the structure of interest. A more detailed explanation about the principles of virtual work and VFM is provided in Appendix C.

2.2. Application of VFM to the atherosclerotic plaque inflation problem

The intraluminal pressurization of atherosclerotic plaques was considered to be a 2D boundary value problem with a plane strain assumption [6]. As this is the first time VFM is applied to the case of plaque intraluminal pressurization, a set of virtual fields (VF) that would allow for the identification of the material stiffness properties had to be defined. The material behavior of the plaque tissue was modeled with a Neo-Hookean hyperelastic material model, as described before [23]. The plaque tissue was deemed to be incompressible. The Cauchy stress for an incompressible Neo-Hookean material model is given as [41]:

$$\sigma_{ij} = -pI_{ij} + 2c_1 B_{ij} \tag{2}$$

Here, c_1 is the material stiffness parameter for the Neo-Hookean material model, I_{ij} is the identity tensor, B_{ij} is the left Cauchy Green deformation tensor, and p is the hydrostatic pressure, which can be nonuniform within the structure. Because the hydrostatic pressure is different for each material point in the structure and its value is not of interest, VF that cancel out the hydrostatic pressure term are desirable. As the hydrostatic pressure only appears on the diagonal of the Cauchy stress tensor, multiplying the stress tensor with a virtual strain tensor with a trace of zero ($tr\left(\frac{\partial \delta u_i}{\partial x_j}\right) = 0$) will remove the hydrostatic pressure in the balance equation (Equation 1). The virtual displacement field with this feature, suitable for this specific application, was identified as:

$$\delta u_i^{[1]} = \begin{bmatrix} \frac{x}{x^2+y^2} \\ \frac{y}{x^2+y^2} \\ 0 \end{bmatrix} \tag{3}$$

where x and y are the coordinates in the deformed configuration, with the origin chosen inside the lumen making sure that the point $x^2 + y^2 = 0$ is not on the inside of the structure as it could lead to infinite values for the VF. The corresponding virtual strain field can be given as:

$$\frac{\partial \delta u_i}{\partial x_j} = \begin{bmatrix} \frac{y^2-x^2}{(x^2+y^2)^2} & \frac{-2xy}{(x^2+y^2)^2} & 0 \\ \frac{-2xy}{(x^2+y^2)^2} & \frac{x^2-y^2}{(x^2+y^2)^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4}$$

Please note that the trace of this virtual strain field is equal to zero. This virtual field is also consistent with the plane strain assumption. Substituting Equations 2, 3 and 4 into Equation 1 and discretizing the structure in N elements with finite dimensions, the balance equation (1) becomes:

$$\begin{aligned} 2c_1 \sum_{e=1}^{N} \left[\left((B_{11} - B_{22}) \cdot \frac{y^2 - x^2}{(x^2+y^2)^2} + 2B_{12} \cdot \frac{-2xy}{(x^2+y^2)^2} \right) \cdot A_e \right] \\ - \sum_{f=1}^{M} \left[\left(\frac{xt_x + yt_y}{x^2+y^2} \right) \cdot L_f \right] = 0 \end{aligned} \tag{5}$$

Here, N is the total number of grid elements of the discretization and, M is the number of line elements on the luminal surface, where the only loading, the intraluminal pressure is applied. A_e is the area of a surface grid element, L_f is the length of a line element on the luminal surface. B_{11}, B_{22}, B_{12}, x, and y, are evaluated at the center of the grid element. t_x and t_y are the x and y components of the traction vector applied on the luminal surface. Equation 5 describes a homogeneous case of atherosclerotic plaque. Since there is only one single material parameter, c_1, this parameter can be directly obtained from the equation by an analytical solution. Plaque tissue is however heterogeneous [2]. The state-of-the-art approach to incorporating the tissue heterogeneity in biomechanical analyses is segmenting the plaque tissue for the mechanically relevant components, namely lipid pool (L), diseased intima (DI), calcified tissue (CT) and the arterial wall (W), where the individual components are deemed to be homogeneous [7]. Equation 5 can be reorganized such that the internal virtual work is obtained for all tissue components separately by integrating over the volume of these components. In the case of four mechanically distinct plaque components, each modeled as incompressible Neo-Hookean material, the discretized balance equation becomes:

$$\begin{aligned} c_{DI} \cdot IVW^*_{DI} + c_L \cdot IVW^*_L + c_{CT} \cdot IVW^*_{CT} + c_W \cdot IVW^*_W \\ - EVW = 0 \end{aligned} \tag{6}$$

Here, IVW^* of one specific component is defined as:

$$IVW^*_{Component} = 2 \sum_{e=1}^{N^*} \left[\left((B_{11} - B_{22}) \cdot \frac{y^2 - x^2}{(x^2+y^2)^2} + 2B_{12} \cdot \frac{-2xy}{(x^2+y^2)^2} \right) \cdot A_e \right] \tag{7}$$

where N^* is the number of elements inside one specific component. A more elaborate explanation is given in Appendix D.

This approach results in an underdetermined system of one equation with four unknowns (c_1 of each component). Theoretically four equations would be sufficient to solve this problem. However, due to the presence of noise, optimization is used rather than a direct analytical solution. For such a minimization approach, having more equations than unknowns can be beneficial. Therefore, new VF were obtained through two means. First, a set of new fields were created by multiplying the first virtual field (Equation 3) with the periodic functions of sin and cosine:

$$\delta \underline{u}^{[2n]} = \delta \underline{u}^{[1]} \cdot cos^{2n}\left(\theta\right) \tag{8}$$

$$\delta \underline{u}^{[2n+1]} = \delta \underline{u}^{[1]} \cdot sin^{2n}\left(\theta\right) \tag{9}$$

Here, $n = 1, 2, 3...$, and θ was defined as the circumference which is related to x and y by:

$$cos^2\left(\theta\right) = \frac{x^2}{x^2 + y^2} \tag{10}$$

and:

$$sin^2\left(\theta\right) = \frac{y^2}{x^2 + y^2} \tag{11}$$

A second set of new VF were generated by using a counter-clockwise rotation (θ_i) of the original coordinate system, defined as:

$$\theta_i = \left(\frac{i}{i_{max}}\right) \cdot \left(180^\circ - \left(\frac{180^\circ}{i_{max} + 1}\right)\right) \tag{12}$$

Here, i is an integer number that ranges from 1 to the desired number of rotations (i_{max}). Rotating the coordinate system results in new independent equations. As the VF are symmetric, the maximum rotation angle was always below 180°. More details on this procedure are given in Appendix D.

Using these two methods, the number of equations can be adjusted to the desired number of equations. These equations then define the set of equations to be used in the optimization procedure to obtain the material parameters for the Neo-Hookean material model. In this study, the Trust Region Reflective (TRR) algorithm available in MATLAB 2017b was utilized. To ensure that the global minimum with the TRR algorithm was found, 100 random initial guesses were made within the search range. The search range for the c_1-values was defined between 0 and 1000 kPa. The mean of the 100 c_1-values, each obtained with a different initial guess, was reported as the estimated c_1-value. A schematic overview of the method is to be found in Appendix D. The MATLAB code that was developed is to be found in Appendix I.

2.3. Validation of the developed VFM-based approach

A validation study was performed to evaluate the performance of the developed VFM-based approach for the material characterization of heterogeneous atherosclerotic plaques. The synthetic experimental measurements were generated from FE simulations (ABAQUS, ver. 2016, Dassault Systèmes) of intraluminal pressurization of ten atherosclerotic plaques.

2.3.1. FE models

The FE plaque geometries were obtained from structural segmentation of histology images of ten human carotid plaques acquired with carotid endarterectomy surgery. Figures for all ten cases are to be found in Appendix E. The final FE plaque models contained the three mechanically relevant components: diseased intima, lipid core and calcified tissue (Figure 1b). The plaques from the endarterectomy surgery do not contain a healthy wall layer. Therefore, a healthy wall layer of 1 mm uniform thickness was added to the FE plaque geometries [42]. For all plaque components, an incompressible Neo-Hookean material model was used. The values for the material parameter c_1 of the components were based on the previous reports [16, 43] i.e.: diseased intima (DI): $c_1 = 120$ kPa, lipid (L): $c_1 = 5$ kPa, calcified tissue (CT): $c_1 = 423$ kPa, and wall (W): $c_1 = 250$ kPa. An intraluminal mean blood pressure of 100 mmHg (13.33 kPa) was applied as the loading condition in the FE simulation. The FE plaque geometries were meshed with plane strain quadrilateral elements with reduced integration with an average edge length of 0.05 mm. The rigid body motions in the simulations were prevented by fully encastering a single node on the outer edge of the healthy wall. The nodal displacements from the FE simulations were extracted as the computational output.

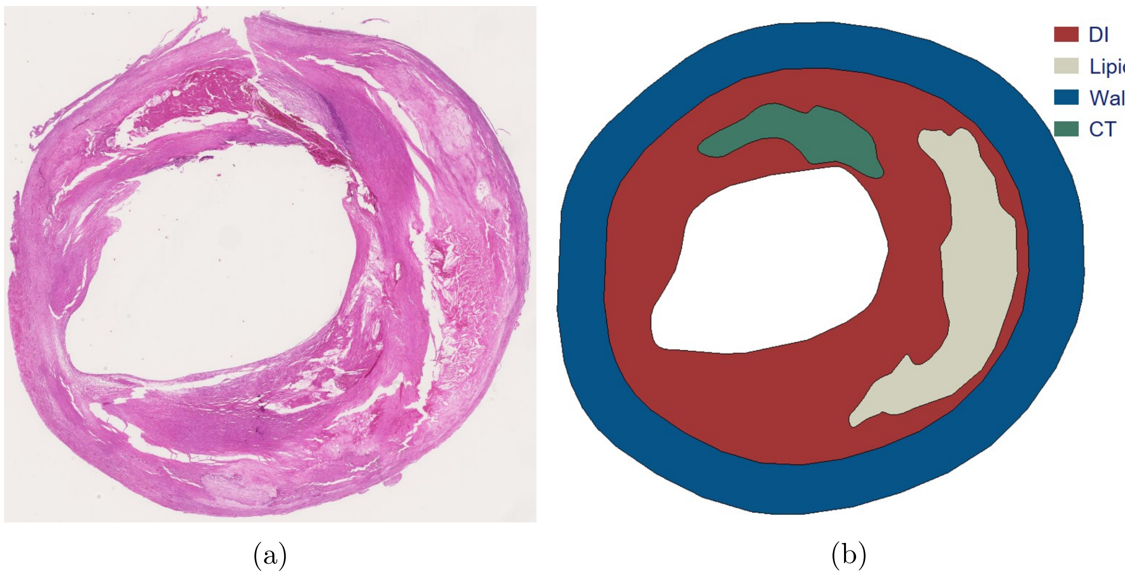


Figure 1: Example of one of the Finite Element Models based on histological slides of human carotid arteries, (a) histological slide and (b) segmented FE geometry Red=Diseased Intima, White=Lipid, Green=Calcified Tissue, Blue=Healthy Wall Tissue.

2.3.2. Synthetic deformation measurements

VFM is suitable for any imaging modality that provides full field deformation data. Such an imaging technique for atherosclerotic plaque deformation measurements is ultrasound imaging [44, 45]. Therefore, in this study the choice was made to simulate plaque deformation data that resembles ultrasound measurements. In this regard, two scenarios were analyzed: (i) an ex-vivo inflation testing case where the plaque deformation was measured with a preclinical, high frequency ultrasound scanner as reported before [34], and (ii) an in-vivo case where the plaque deformation was measured with a clinical scanner [46]. The preclinical ultrasound scanner was considered to have a resolution of 15 μm x 55 μm (axial x lateral) [34] and the clinical ultrasound scanner was considered to have a resolution of 75 μm x 275 μm [46]. The axial direction is defined as the direction along the ultrasound beams and the lateral direction was defined perpendicular to that.

The FE nodal displacements in higher resolution were down-sampled and interpolated to two grids, one to mimic the typical measurements of a 2D atherosclerotic cross-section with the preclinical ultrasound scanner and the other one with the clinical scanner with the aforementioned, corresponding resolutions. The translation of the simulated plaque deformation to the grids and derivation of corresponding full-field strain in the plaque structure was performed through a custom-built MATLAB code (Mathworks, ver. 2017a). Further details of this procedure can be found in Appendix E.2 and Appendix I.

2.3.3. Material characterization with the developed VFM and impact of noise

The synthetic plaque deformation measurements were then used as input for the proposed VFM-based approach. The estimated values for the c_1 material parameter for the Neo-Hookean models of the components were then compared to the ground-truth values that were set as input to the FE model.

As there is always noise present in both ex-vivo and in-vivo real-life ultrasound measurements, the influence of noise on the developed VFM-based approach was evaluated as well. The expected accumulated level of noise at 100 mmHg for the clinical ultrasound scanner can be represented by a Root Mean Square Error (RMSE) 3% of the displacement in the axial direction, and 9% of the displacement in the lateral direction [46]. For the high frequency, preclinical scanner, the noise level will scale linearly with the resolution of the scanner. The expected accumulated level of noise at 100 mmHg for the high frequency scanner was therefore represented by a RMSE 0.6% of the displacement in the axial direction, and 1.8% of the displacement in the lateral direction [34, 46]. The c_1-values were recalculated using the noise data. More details on the level of noise used can be found in Appendix E.3.

2.4. Optimal number of fields

VFM, as proposed in this study, can be used to obtain an arbitrary number of equations (See section 2.2). An analysis was performed to evaluate the optimal number of the equations. For this purpose, a grid search procedure was employed where the numbers of VF used in the VFM-based approach varied by using different combinations of rotation numbers and the degree of the periodic functions. A maximum number of 50 rotations and a maximum degree of 20 for the periodic functions were used. This analysis was performed both for the clinical and preclinical ultrasound scanner settings. A more elaborate explanation can be found in Appendix F.

2.5. Application to real data

Lastly, the developed VFM-based approach was applied to an illustrative case of real tissue deformation measurements. An already available full-field plaque deformation data from an ex-vivo inflation experiment with an excised atherosclerotic human coronary was used for this purpose [47]. The procedure for this inflation experiment was similar to the procedure used previously with atherosclerotic iliac arteries from diabetic pigs [34]. For the current study, the plaque deformation data at 100 mmHg, collected with a high frequency ultrasound scanner was used. The plaque cross-section structural segmentation was based on magnetic resonance imaging [47]. The atherosclerotic artery cross-section contained a healthy wall, diseased intima and a lipid core. The c_1-values for the Neo-Hookean material model of the 3 components were estimated with the developed VFM-based approach.

3. Results

3.1. Validation study

3.1.1. Illustrative example

The proposed VFM-based approach for the heterogeneous material characterization of atherosclerotic plaques from intraluminal inflation measurements was successfully developed and applied on ten plaque cross-sections with varied geometries. The results of a representative plaque cross-section (Figure 1) are reported here. For this cross-section, a grid with a resolution of 10μm x 10μm, and 6 VF together with 50 rotations , resulting in a set of 300 objective functions to be utilized in the minimization, were used. The minimization scheme was run for 100 random initial guesses. The mean and standard deviation ($\pm$ SD) of the estimated c_1-values for the components are reported in Table 1. The estimated c_1-value (mean $\pm$ SD) was 118.9 kPa $\pm$ 0.9 kPa for the diseased intima, 7.6

Table 1: Estimated c_1-values for all four components, using the mean of 100 random initial guesses in the "Trust Region Reflective" algorithm.

	DI	**L**	**CT**	**W**
Ground truth (kPa)	120	5	423	250
Estimated c_1 (kPa)	118.9 $\pm$ 0.9	7.6 $\pm$ 1.3	416.4 $\pm$ 1.9	230 $\pm$ 6.1

kPa $\pm$ 1.3 kPa for the lipid, 416.4 kPa $\pm$ 1.9 kPa for the calcified tissue, and 230.0 kPa $\pm$ 6.1 kPa for the wall tissue, the relative error was the smallest for the diseased intima tissue (0.9%) and the largest for the lipid (52%). The absolute error was the smallest for the diseased intima tissue (1.1 kPa) and the largest for the wall tissue (20 kPa).

Optimal number of virtual fields and rotations

To obtain an estimate for the optimal combination of VF and rotations, c_1-values were estimated for different combinations of VF and rotations. For this study, the optimal combination was determined based on the relative error for the diseased intima component.

Rupture of the diseased intima component specifically the fibrous cap overlying a lipid pool, is the main cause of cardiovascular events [4]. It is therefore considered to be more important to estimate these material stiffness properties accurately than the other components in order to achieve better stress analysis results [7].

Figure 2 shows the relative error between the estimated c_1-values and the ground truth values for different combinations, using the high frequency scanner measurements. A maximum number of 20 VF was tested, where larger numbers of VF were obtained by

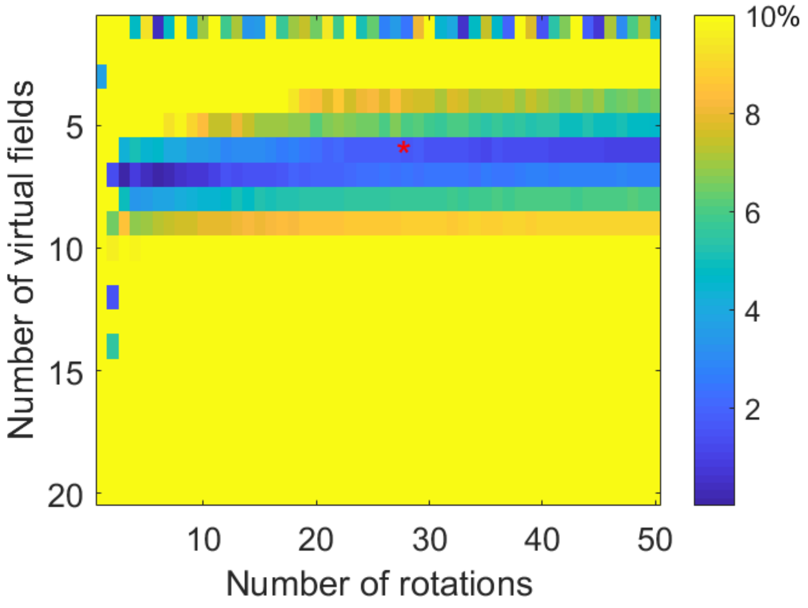


Figure 2: Relative error between the estimated c_1-values and the ground truth value for different combinations for case 8, using the high frequency scanner measurements, the color-code represents the magnitude of the relative error.

increasing the order of the trigonometric functions, as explained in section 2.2 (See also Appendix C):

$$\delta \underline{u}^{[N]} = \left[\delta \underline{u}^{[1]} \cdot cos^2(\theta); \delta \underline{u}^{[1]} \cdot sin^2(\theta); \delta \underline{u}^{[1]} \cdot cos^4(\theta); ...\right]^T \tag{13}$$

Here, N is the number of VF, also reflecting the number of components in the vector. Color-coded Figure 2 reports the magnitude of the relative error. (Please note that the legend was adjusted such that relative error values $\geq$10% are indicated in yellow). Although there were few combinations outside the following regions, the majority (83%) of the combinations with <10% error were located roughly between 4-9 VF and 3-50 rotations. The combination with the least relative error (0.01%) was 7 VF and 3 rotations.

The optimal combination was defined as the median of combinations with a relative error <10%. For this illustrative example, the optimal combination was achieved at 6 VF and 27 rotations (Figure 2, red asterisk) resulting in 162 objective functions. The estimated c_1-values for this case were: diseased intima: 122 kPa, lipid: 8.8 kPa, calcified tissue: 416 kPa, and wall: 217 kPa.

3.1.2. Overall results

For all 10 plaque cross-sections, there were a set of combinations of VF and rotations that presented an error smaller than 10%. The minimum error for the cross-sections varied from 0.0005% to 0.4%. For all cross-sections the trend was similar to the illustrative case (Figure 2), where there was a smaller margin of the number of VF and rotations resulting in c_1-values with less than 10% error. The results for all cases are shown in Figure 3. Table

Table 2: Optimal combinations of VF and rotations, using the median of the combinations that give a relative error lower than 10%, for the high frequency scanner resolution.

Case:	1	2	3	4	5	6	7	8	9	10	**Median**
VF:	6	8	7	6	7	7	7	6	9	5	7
Rot:	28	28	26	25	26	25	26	27	25	22	26

2 presents the optimal combinations for all cases for the high frequency scanner. For the results of all cases combined, the range for the number of VF was from 5 to 9 and for the rotations from 22 to 28. The median value for the optimal combination of all cases was 7 VF and 26 rotations. This combination was defined as the optimal combination of VF and rotations for the high frequency scanner resolution. For the clinical scanner resolution, a similar trend for the error map was obtained. However, obtaining results with <10% error required more VF in general. For one case (case #1), the minimum error was 14%, where the minimum error for the other 9 cases ranged from 0.16 to 2.4%. For this one case the threshold was therefore adjusted to 25% to be able to make an estimation of the optimal combination of VF and rotations. Cases #1, #5, and #10 showed a smaller number of combinations with <10% error. The remainder of the cases showed a comparable number of combinations with <10% error to the number of combinations of the high frequency scanner results. The results for all cases are shown in Figure 4. Table 3 presents the optimal combinations for all cases for the clinical scanner. The optimal number of VF

Table 3: Optimal combinations of VF and rotations, using the median of the combinations that give a relative error lower than 10%, for the clinical scanner resolution. For case #1 a threshold of 25% was used.

Case:	1	2	3	4	5	6	7	8	9	10	**Median**
VF:	1	17	15	5	4	9	8	11	12	1	9
Rot:	17	27	26	26	2	27	28	26	26	28	26

ranged from 1 to 17, and the optimal number of rotations ranged from 17 to 28. The median value for the optimal combination was 9 VF and 26 rotations. If the cases (#1, #5, and #10) with a relatively smaller number of combinations resulting in <10% were excluded, the range for the VF reduced to [5 - 17] and for rotations to [26 - 28], with the

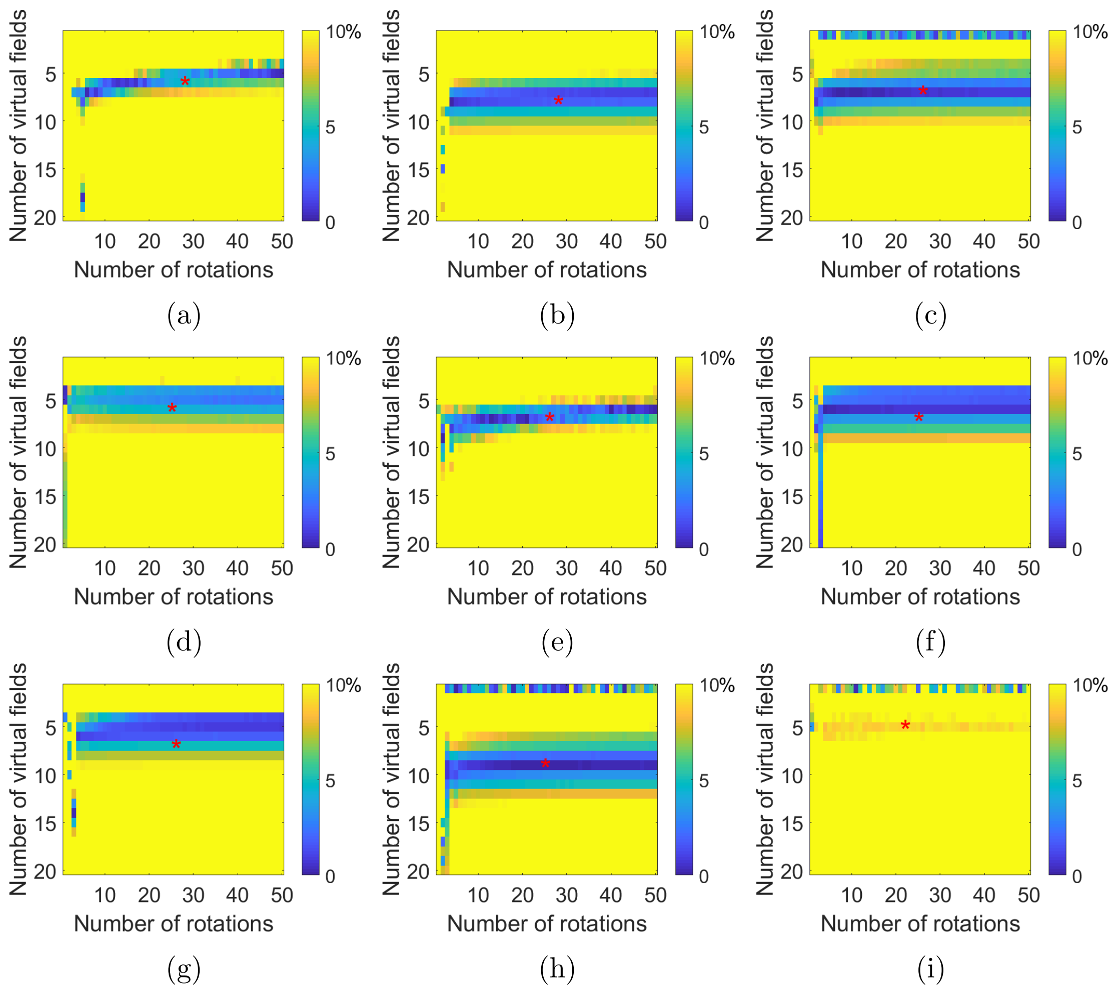


Figure 3: Relative error for different combinations of VF and rotations, for the high frequency scanner resolution, (a) case 1, (b) case 2, (c) case 3, (d) case 4, (e) case 5, (f) case 6, (g) case 7, (h) case 9 and (i) case 10.

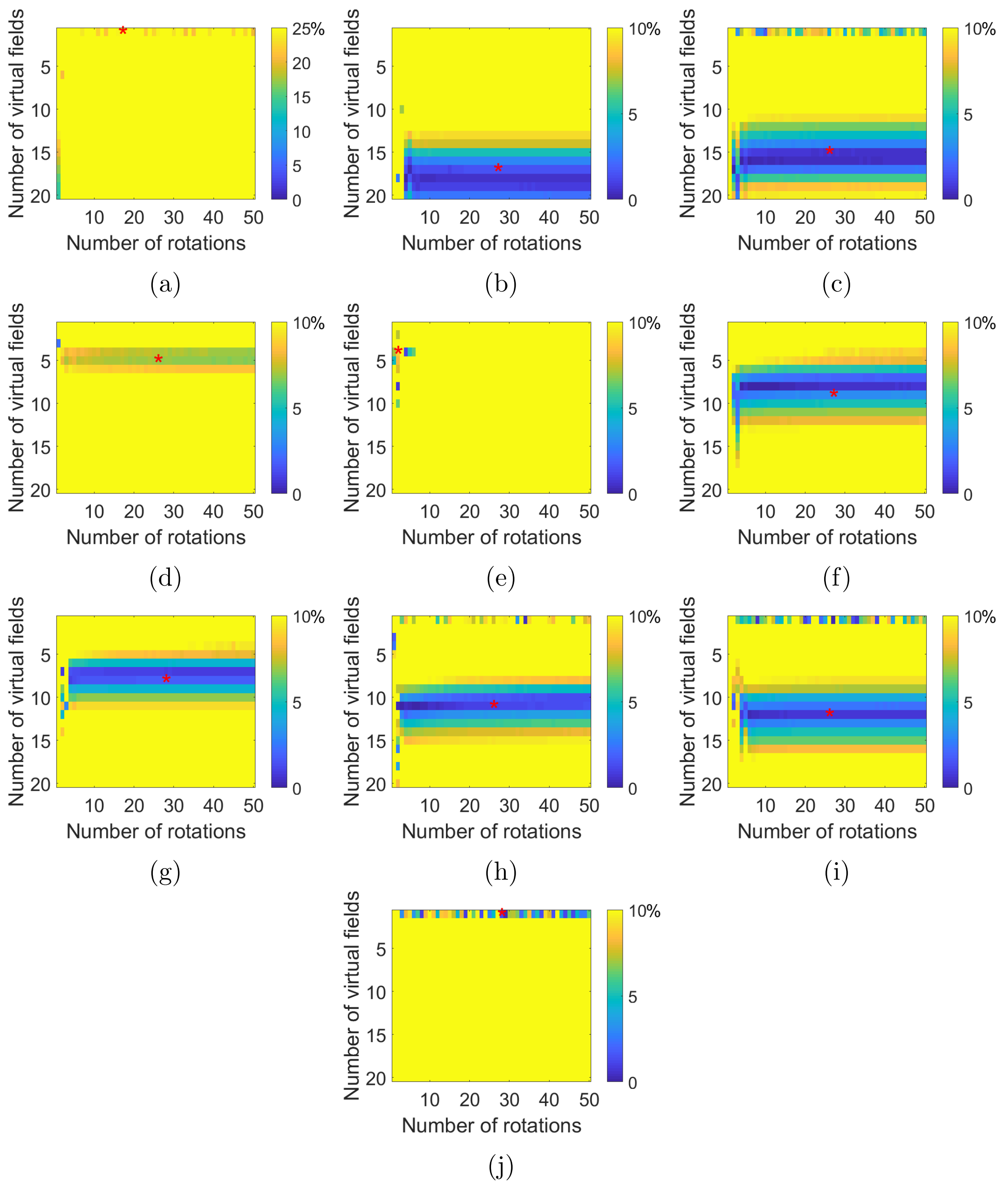


Figure 4: Relative error for different combinations of VF and rotations, for the clinical scanner resolution, (a) case 1, (b) case 2, (c) case 3, (d) case 4, (e) case 5, (f) case 6, (g) case 7, (h) case 8, (i) case 9 and (j) case 10.

median values for the optimal combination being 11 VF and 26 rotations. The median of all, 9 VF and 26 rotations, was used as the optimal combination of VF and rotations in further analyses.

Using the optimal number of 7 VF and 26 rotations, as reported in Table 2, the c_1-values of all four components were calculated. Table 4 presents the estimated c_1-values for the high frequency scanner setting.

Table 4: Estimated c_1-values for the high frequency scanner resolution, using the optimal combination of 7 VF and 26 rotations (Table 2), *When excluding the three cases in which the relative area of the calcified tissue was <2% of the total area.

	DI	**L**	**CT**	**W**
Ground Truth	120	5	423	250
Case 1	110	2	488	257
Case 2	122	4	372	240
Case 3	119	12	840	190
Case 4	112	13	394	250
Case 5	117	4	505	176
Case 6	116	4	391	245
Case 7	114	4	1134	230
Case 8	117	8	400	213
Case 9	127	9	596	207
Case 10	105	1	5641	253
Mean ± SD	116 ± 6	6 ± 4	1077 ± 1622	226 ± 28
Mean* ± SD*	119 ± 5	8 ± 4	500 ± 171	217 ± 28

The mean (± SD) estimated c_1-values for the diseased intima, lipid, calcified tissue, and wall were 116 (± 6) kPa, 6 (± 4) kPa, 1077 (± 1622) kPa, and 226 (± 28) kPa, respectively. Estimations of all four components were in good agreement with the ground truth values, except for the calcified tissue. In three of the ten cases the relative area of the calcified tissue was <2% of the total area (case #1, #7, and #10). The mean estimated c_1-value of the calcified tissue for the other seven cases was 500 kPa, which is in better agreement with the ground-truth value, and demonstrated a smaller SD (171 kPa) (Table 4).

Using the optimal number of 9 VF and 26 rotations, the c_1-values of all four components were calculated for the clinical scanner setting. Table 5 presents the estimated c_1-values for the clinical scanner setting. The estimated c_1-values (mean ± SD) for the diseased intima, lipid, calcified tissue, and wall were: 107 kPa ± 31 kPa, 5 kPa ± 8 kPa, 1705 ± 2905 kPa, and 288 kPa ± 144 kPa. When excluding the cases #1, #7, and #10 with a small calcified area, the estimated c_1-values (mean ± SD) for the diseased intima, lipid, calcified tissue, and wall, were: 120 kPa ± 23 kPa, 7 kPa ± 9 kPa, 585 kPa ± 602 kPa, and 284 kPa ± 167 kPa.

The time required for the estimation of the c_1-values using the optimal combination of VF and rotations was (mean ± SD) 14 (± 3) seconds for the high frequency scanner

Table 5: Estimated c_1-values for the clinical scanner resolution, using the optimal combination of 9 VF and 26 rotations (Table 3), *When excluding the three cases in which the relative area of the calcified tissue was <2% of the total area.

	DI	**L**	**CT**	**W**
Ground Truth	120	5	423	250
Case 1	48	0.003	5	397
Case 2	146	1	277	259
Case 3	138	3	1886	114
Case 4	102	13	251	355
Case 5	80	0.1	187	616
Case 6	116	0.7	384	273
Case 7	115	3	3600	203
Case 8	126	3	361	156
Case 9	129	25	749	214
Case 10	75	0.1	9353	298
Mean ± SD	107 ± 31	5 ± 8	1705 ± 2905	288 ± 144
Mean* ± SD*	120 ± 23	7 ± 9	585 ± 602	284 ± 167

resolution and 6 (± 1) seconds for the clinical scanner resolution. Analyses were performed on a laptop pc with an Intel i7-7700HQ processor at 2.80 GHz with 16 GB of RAM.

Influence of the measurement noise

Table 6 shows the mean (± SD) of the estimated c_1-values with 100 random noise cases for the high frequency scanner setting. The noise in the measurements lowered the accuracy of the estimation of the c_1-values. The estimated c_1-value (mean ± SD) was 114 kPa ± 11 kPa for the diseased intima, 8 kPa ± 6 kPa for the lipid, 947 kPa ± 1315 kPa for the calcified tissue, and 246 kPa ± 52 kPa for the wall. When excluding the cases #1, #7, and #10 with a small calcified area, the estimated c_1-values (mean ± SD) for the diseased intima, lipid, calcified tissue, and wall in the presence of noise were: 112 kPa ± 8 kPa, 8 kPa ± 4 kPa, 433 kPa ± 183 kPa, and 255 kPa ± 57 kPa, respectively. Compared to the case without noise, the estimations for all components except the calcified tissue got slightly worse.

Table 7 shows the mean (± SD) of the estimated c_1-values with 100 random noise cases for the clinical scanner setting. The estimated c_1-value (mean ± SD) was 107 kPa ± 30 kPa for the diseased intima, 6 kPa ± 9 kPa for the lipid, 1708 kPa ± 3037 kPa for the calcified tissue and 285 kPa ± 135 kPa for the wall. When excluding the cases with a small calcified area, the estimated c_1-values (mean ± SD) for the diseased intima, lipid, calcified tissue, and wall, in the presence of noise were: 119 kPa ± 21 kPa, 8 kPa ± 10 kPa, 591 kPa ± 565 kPa, and 280 kPa ± 152 kPa, respectively. In general, the estimations for the clinical scanner resolution and noise decreased the accuracy of the c_1 estimation. This is mainly indicated by the increased standard deviation.

Table 6: Estimated values of c_1 for the high frequency scanner with realistic noise, Mean ± SD, *When excluding the three cases, in which the relative area of the calcified tissue was <2% of the total area.

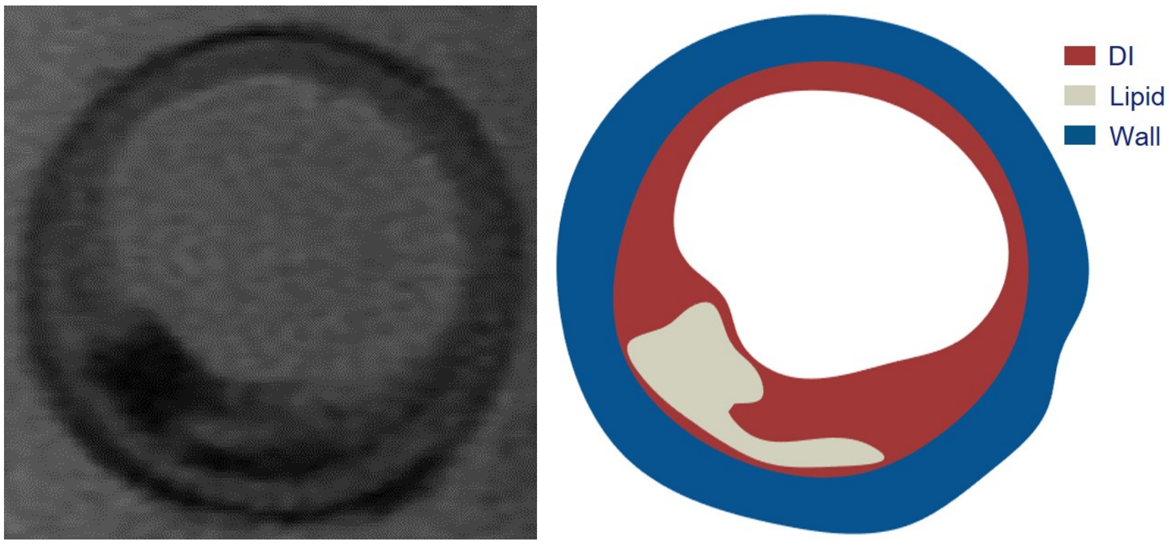
	DI	**L**	**CT**	**W**
Ground Truth	120	5	423	250
Case 1	135 ± 10	3 ± 1	859 ± 131	195 ± 25
Case 2	117 ± 1	4 ± 1	384 ± 4	239 ± 2
Case 3	119 ± 1	12 ± 3	822 ± 85	192 ± 8
Case 4	108 ± 1	8 ± 2	373 ± 18	265 ± 8
Case 5	107 ± 3	2 ± 1	257 ± 85	368 ± 55
Case 6	96 ± 3	8 ± 1	282 ± 45	268 ± 8
Case 7	115 ± 1	5 ± 1	948 ± 133	231 ± 2
Case 8	115 ± 1	11 ± 1	408 ± 3	218 ± 4
Case 9	121 ± 2	8 ± 2	504 ± 33	232 ± 6
Case 10	102 ± 2	23 ± 3	4634 ± 1250	248 ± 7
Mean ± SD	114 ± 11	8 ± 6	947 ± 1315	246 ± 52
Mean* ± SD*	112 ± 8	8 ± 4	433 ± 183	255 ± 57

Table 7: Estimated values of c_1 for the clinical scanner with realistic noise, Mean ± SD, *When excluding the three cases, in which the relative area of the calcified tissue was <2% of the total area.

	DI	**L**	**CT**	**W**
Ground Truth	120	5	423	250
Case 1	48 ± 2	0.01 ± 0.002	7 ± 8	396 ± 6
Case 2	145 ± 3	2 ± 0.5	278 ± 8	258 ± 4
Case 3	138 ± 1	4 ± 3	1892 ± 186	111 ± 11
Case 4	101 ± 2	17 ± 10	257 ± 39	352 ± 22
Case 5	81 ± 5	0.3 ± 0.3	215 ± 84	595 ± 71
Case 6	115 ± 6	1 ± 1	380 ± 101	276 ± 25
Case 7	115 ± 2	3 ± 1	3598 ± 352	201 ± 8
Case 8	127 ± 1	4 ± 1	362 ± 6	152 ± 8
Case 9	129 ± 7	25 ± 6	754 ± 94	213 ± 23
Case 10	75 ± 9	0.2 ± 0.2	9340 ± 4060	298 ± 19
Mean ± SD	107 ± 30	6 ± 9	1708 ± 3037	285 ± 135
Mean* ± SD*	119 ± 21	8 ± 10	591 ± 565	280 ± 152

3.2. Case study: material properties of human coronary plaque from real ex-vivo inflation measurements

The developed VFM-based approach was applied to characterize the material stiffness properties of a human coronary plaque using real ex-vivo inflation measurements. The MRI-based segmentation of the plaque cross-section is shown in Figure 5. The cross-section presented contains a diseased intima, a lipid pool and healthy wall, but no calcified tissue. The c_1-values of the three components were estimated using 7 VF with 26 rotations



(a) (b)

Figure 5: (a) MRI image of the plaque cross-section and (b) segmentation based on MRI. Red: Diseased Intima (DI), White: Lipid (L), Blue: Wall (W).

.

as it was obtained as the optimal combination from the validation study (Section 3.1.2). As the experimental deformation data was available at multiple pressure steps, c_1-values were estimated using the measurements from two pressure steps: 60 and 100 mmHg. Table 8 presents the estimated c_1-values for the three plaque components. The estimated

Table 8: Estimated c_1 values for the three components in kPa, at two different pressure steps.

	DI	**L**	**W**
60 mmHg	15.1	9.1	17.9
100 mmHg	21.5	13.3	23.6

c_1-value for the diseased intima increased from 15.1 kPa to 21.5 kPa, for the lipid from 9.1 kPa to 13.3 kPa, and for the wall from 17.9 kPa to 23.8 kPa with increasing pressure. The wall was the stiffest component, whereas the lipid was the most compliant. The VFM estimations took approximately 4 min.

4. Discussion

In this study VFM was used, for the first time, for the material characterization of atherosclerotic plaques. First, the VFM-based technique was developed that enables the estimation of heterogeneous material stiffness properties of atherosclerotic arteries from inflation deformation measurements. For this technique, appropriate virtual fields were identified, a minimization scheme was defined, and the required numerical analysis was done to get the material stiffness properties from the minimization problem. Subsequently, a validation study was performed to evaluate the performance of the developed technique. For this purpose, synthetic experimental deformation measurements were obtained from FE simulations of intraluminally pressurized atherosclerotic plaques (n=10). The performance of the developed technique was evaluated for two scenarios. In the first scenario, the synthetic deformation maps mimicked the measurements obtained with a high frequency ultrasound scanner (e.g Vevo 2100, linear probe with 40 MHz central frequency [34]), a possible case for ex-vivo inflation experiments. In the second scenario, the synthetic deformation maps mimicked the carotid artery measurements from clinical ultrasound scanners (e.g A Philips SONOS 7500, linear probe with 8.7 MHz central frequency [46]). The influence of the possible noise in the measurements was determined as well. Finally, a case study was performed using the proposed VFM-based approach on real experimental inflation data of a coronary artery.

4.1. Validation study

VFM requires prescribing kinematically admissible virtual displacement fields. Although various choices can be made, in the current study, these fields were obtained by using a defined field together with periodic functions and rotations of the original coordinate system.

The sensitivity analyses of the influence of the number of VF and rotations demonstrated that the range of the number of the VF was much smaller than for the rotations for the high accuracy solutions, defined as the ones with <10% error for the diseased intima material stiffness estimation. The optimal solution was defined as the combination of the median for both parameters. Across the ten plaque cross-sections analyzed, for the high frequency ultrasound setting, the optimal number of VF ranged from 5 to 9, and the optimal number of rotations ranged from 22 to 28. For the clinical scanner resolution, this range for the number of VF changed to 1 to 17. An interesting finding was that three out of the 10 cases (case #1, #5 and #10) were more sensitive to the resolution of the deformation measurements demonstrated by the smaller number of combinations of optimal solutions in the clinical resolution setting. This effect is most likely caused by geometrical factors as these are the only differences between the 10 cases. Another important finding was that for the other seven cases more VF were required to make c_1 estimations for the diseased intima with <10% error. This suggests that the optimal combinations of VF and rotations depend on the resolution of the deformation measurements used.

In this validation study, ground truth c_1-values were known, hence the optimal combination of VF and rotations were based on the relative error between the estimations and the ground truth values. In a real life scenario, where the ground truth value is unknown, the optimal solution might be calculated from the gradient of the estimated c_1-values. As there are clear ranges of combinations of VF and rotations in which the relative error is low, by using a gradient based approach the best estimation of the material stiffness value can be obtained. Such an approach requires further research.

Using the optimal number of VF and rotations (7 VF and 26 rotations for high frequency ultrasound scanner resolution, and 9 VF and 26 rotations for the clinical scanner resolution) based on the relative error of the estimated c_1-value for the diseased intima, the c_1-values for all four components were estimated. This approach yielded successful estimations of the diseased intima, lipid, and wall tissue for the high frequency scanner setting in all the ten cases, and also for the calcified tissue in six cases. However, for three cases the obtained c_1-value for the calcified tissue was up to 13 times greater than the ground truth value. These cross-sections were the ones with a calcified tissue area $<2\%$ of the total area. For one case (Case #3), the estimated value for the calcified tissue was twice the ground truth value. For this single case, the calcified tissue area was 7% of the total area. There was however a large lipid pool between the calcified area and the lumen. The error in these four cases can be explained as follows. VFM is a method based on energy balance principle. The total strain energy stored inside the small components is relatively small. Therefore, the minimization problem is less sensitive to changes in the c_1-value estimation of components with a small area, and hence more prone to error in the c_1-value estimations for the small components. Similarly, case #4 with a small lipid component showed approximately 160% error in the c_1-value estimation for lipid pool whereas c_1-values for the other components were very successful. This also explains the error for the one case with a calcified tissue area of 7%, as the large lipid pool between the lumen and the calcified area will decrease the strain energy stored in the calcified area.

This effect is even more amplified if the small component is away from the lumen, such as for case #10. Because of the choice of the VF made in this study, the weight that a component gets from the VF is inversely correlated to its distance to the lumen center (see Equation 5). Moreover, because the pressure is applied intraluminally, components further away from the lumen show deformations smaller than the ones located close to the lumen. Therefore, a limitation for the method as proposed in this study is that the small components, especially if not located close to the lumen, are more prone to error for material stiffness estimation. These components are however also the ones that have much less influence on the stress distribution in the diseased tissue, where the rupture leading to clinical events occurs [4]. If required, the accuracy of the proposed VFM-based technique for the components that are away from the lumen or small in size the proposed method can be improved by using virtual fields, e.g obtained by a translating coordinate system.

Due to the lower resolution of the deformation maps, the accuracy of c_1 estimations for the clinical scanner setting was lower, as observed by the larger standard deviations. Nevertheless the material stiffness estimations of the diseased intima, the most important component, lipid and wall were successful in most cases.

When including realistic noise for the high frequency scanner setting, the estimated c_1-values were in good agreement with the ground truth values for the diseased intima, lipid and wall. When excluding the three cases with small calcified regions, also the estimated c_1-values were in good agreement with the ground truth values. The standard deviation was the smallest for the lipid and diseased intima. Relative to the mean value, the standard deviation was the smallest for the diseased intima, indicating a more accurate estimation.

For the clinical scanner setting with noise, estimated c_1-values were less accurate, indicated by the larger standard deviations. The diseased intima was estimated more accurately than the other components, and was still in good agreement with the ground truth values in most cases.

4.2. Case study: material properties of human coronary plaque from real ex-vivo inflation measurements

The demonstration of the developed method with the real ultrasound measurement data was performed with ex-vivo inflation measurements of an atherosclerotic human coronary artery. The estimated material stiffness values were in agreement with the reported literature data [27, 29], as shown in Figure 6. In the figure, the c_1-value obtained

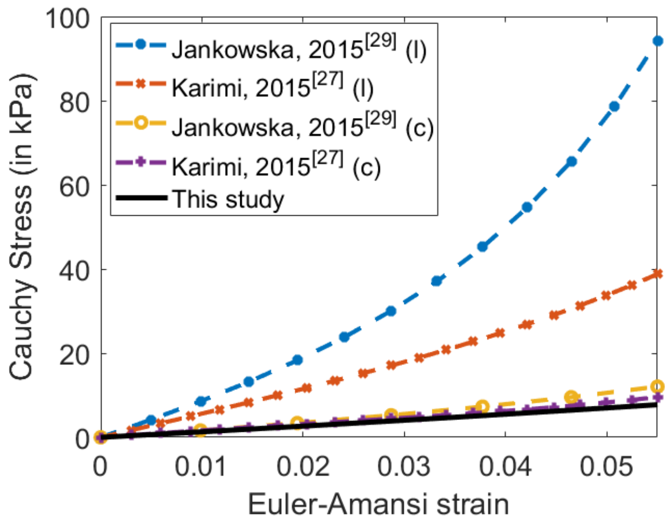


Figure 6: The Cauchy stress at a certain level of uniaxially applied Euler-Amansi strain, calculated using the material coefficients in the constitutive relations reported by Karimi et al. [27], and Jankowska et al. [29], (c) circumferential loading, and (l) longitudinal loading.

at 100 mmHg was used to present the material behavior estimated with the developed VFM-based technique. Our results match well with the material behavior obtained with the uniaxial tensile tests performed in the circumferential direction of the plaque rather than the longitudinal direction [27, 29]. It is important to note that during inflation tests the tissue is mainly stretched in the circumferential direction.

The reported c_1-values for the plaque components increased over the different pressure steps (Table 8). This suggests that the Neo-Hookean material model is limited in capturing the nonlinear behavior of the tissue. Although Neo-Hookean material models were employed in this study, the developed VFM-based approach can be used with any material model such as a Mooney-Rivlin material model [20] or a Yeoh material model [18].

4.3. Implications for in-vivo application

An important motivation for developing the VFM-based approach was the large computational time required for other techniques, such as inverse FE modelling. The computational time required for the VFM-based approach is in the order of seconds and minutes whereas the alternative techniques might require hours or even days to estimate the material stiffness properties. The required time for VFM can be even more reduced by using faster minimization algorithms or coding the procedure more effectively.

The VFM-based approach can be easily extended to estimate the material stiffness of a larger number of components. This will require the studies on the size limitation of the components observed in this study. Parametric studies can be performed to obtain the estimates of the minimal size of components for accurate stiffness assessment. If the current method is extended sufficiently it can also potentially be used to perform an element by element material stiffness estimation. With this approach there will be no need for prior segmentation of the plaque components, leading to a segmentless heterogeneous material characterization technique.

5. Conclusion

In this study, VFM was applied for the material characterization of atherosclerotic plaques for the first time. The developed VFM-based technique enables the characterization of heterogeneous plaque tissue from full-field inflation measurements. The developed technique was validated using synthetic experimental results for a potential source of plaque deformation measurements i.e., the ultrasound imaging measurements. Although higher resolution deformation maps with smaller noise levels were shown to provide more accurate results, the VFM-based technique demonstrated good performance for both the high frequency and clinical ultrasound scanner settings tested. Moreover, the developed VFM-based technique was successfully applied to real experimental data for human coronary plaque characterization. The obtained results showed good agreement with the previously reported coronary plaque material properties.

Compared to existing material characterization methods, such as inverse FEM, the developed VFM-based technique is more attractive as it is computationally less expensive. The full-field deformation maps required for the technique can be obtained from various medical imaging modalities including ultrasound and MRI. As such, it has a great potential to be used for plaque specific material characterization from in-vivo deformation measurements. Moreover, the developed VFM-based technique has the potential to be extended for material characterization of more plaque components than employed in the current study, which warrants further studies.

6. References

[1] R. Lozano, M. Naghavi, K. Foreman, S. Lim, K. Shibuya, V. Aboyans, J. Abraham, T. Adair, R. Aggarwal, S. Y. Ahn, et al., Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a systematic analysis for the global burden of disease study 2010, The lancet 380 (2012) 2095–2128.

[2] E. Falk, Pathogenesis of atherosclerosis, Journal of the American College of Cardiology 47 (2006) C7–C12.

[3] T. G. Brott, J. L. Halperin, S. Abbara, J. M. Bacharach, J. D. Barr, R. L. Bush, C. U. Cates, M. A. Creager, S. B. Fowler, G. Friday, et al., Guideline on the management of patients with extracranial carotid and vertebral artery disease, Journal of the American College of Cardiology 57 (2011) e16–e94.

[4] A. Zaman, G. Helft, S. Worthley, J. Badimon, The role of plaque rupture and thrombosis in coronary artery disease, Atherosclerosis 149 (2000) 251–266.

[5] A. C. Akyildiz, L. Speelman, H. A. Nieuwstadt, H. van Brummelen, R. Virmani, A. van der Lugt, A. F. van der Steen, J. J. Wentzel, F. J. Gijsen, The effects of plaque morphology and material properties on peak cap stress in human coronary arteries, Computer Methods in Biomechanics and Biomedical Engineering 19 (2016) 771–779.

[6] G. C. Cheng, H. M. Loree, R. D. Kamm, M. C. Fishbein, R. T. Lee, Distribution of circumferential stress in ruptured and stable atherosclerotic lesions. a structural analysis with histopathological correlation., Circulation 87 (1993) 1179–1187.

[7] A. C. Akyildiz, L. Speelman, H. van Brummelen, M. A. Gutiérrez, R. Virmani, A. van der Lugt, A. F. van der Steen, J. J. Wentzel, F. J. Gijsen, Effects of intima stiffness and plaque morphology on peak cap stress, Biomedical Engineering Online 10 (2011) 25.

[8] C. Zarins, S. Glagov, Stress analysis of the diseased arterial cross-section, Advances in Bioengineering 2 (1990) 273.

[9] A. Veress, J. F. Cornhill, K. Powell, E. Herderick, J. Thomas, Finite element modeling of atherosclerotic plaque, in: Computers in Cardiology 1993, Proceedings., IEEE, pp. 791–794.

[10] R. T. Lee, A. J. Grodzinsky, E. H. Frank, R. D. Kamm, F. J. Schoen, Structure-dependent dynamic mechanical behavior of fibrous caps from human atherosclerotic plaques., Circulation 83 (1991) 1764–1770.

[11] R. T. Lee, S. G. Richardson, H. M. Loree, A. J. Grodzinsky, S. A. Gharib, F. J. Schoen, N. Pandian, Prediction of mechanical properties of human atherosclerotic

tissue by high-frequency intravascular ultrasound imaging. an in vitro study., Arteriosclerosis, Thrombosis, and Vascular Biology 12 (1992) 1–5.

[12] H. M. Loree, A. J. Grodzinsky, S. Y. Park, L. J. Gibson, R. T. Lee, Static circumferential tangential modulus of human atherosclerotic tissue, Journal of Biomechanics 27 (1994) 195–204.

[13] G. A. Holzapfel, G. Sommer, P. Regitnig, Anisotropic mechanical properties of tissue components in human atherosclerotic plaques, Journal of Biomechanical Engineering 126 (2004) 657–665.

[14] R. A. Baldewsing, J. A. Schaar, F. Mastik, C. W. Oomens, A. F. van der Steen, Assessment of vulnerable plaque composition by matching the deformation of a parametric plaque model to measured plaque deformation, IEEE Transactions on Medical Imaging 24 (2005) 514–528.

[15] S. Barrett, M. Sutcliffe, S. Howarth, Z.-Y. Li, J. Gillard, Experimental measurement of the mechanical properties of carotid atherothrombotic plaque fibrous cap, Journal of Biomechanics 42 (2009) 1650–1655.

[16] D. M. Ebenstein, D. Coughlin, J. Chapman, C. Li, L. A. Pruitt, Nanomechanical properties of calcification, fibrous tissue, and hematoma from atherosclerotic plaques, Journal of Biomedical Materials Research Part A 91 (2009) 1028–1037.

[17] E. Maher, A. Creane, S. Sultan, N. Hynes, C. Lally, D. J. Kelly, Tensile and compressive properties of fresh human carotid atherosclerotic plaques, Journal of Biomechanics 42 (2009) 2760–2767.

[18] M. G. Lawlor, M. R. O'Donnell, B. M. O'Connell, M. T. Walsh, Experimental determination of circumferential properties of fresh carotid artery plaques, Journal of Biomechanics 44 (2011) 1709–1715.

[19] E. Maher, A. Creane, S. Sultan, N. Hynes, C. Lally, D. J. Kelly, Inelasticity of human carotid atherosclerotic plaque, Annals of Biomedical Engineering 39 (2011) 2445–2455.

[20] M. H. Kural, M. Cai, D. Tang, T. Gwyther, J. Zheng, K. L. Billiar, Planar biaxial characterization of diseased human coronary and carotid arteries for computational modeling, Journal of Biomechanics 45 (2012) 790–798.

[21] V. M. Heiland, C. Forsell, J. Roy, U. Hedin, T. C. Gasser, Identification of carotid plaque tissue properties using an experimental–numerical approach, Journal of the Mechanical Behavior of Biomedical Materials 27 (2013) 226–238.

[22] A. Karimi, M. Navidbakhsh, A. Shojaei, S. Faghihi, Measurement of the uniaxial mechanical properties of healthy and atherosclerotic human coronary arteries, Materials Science and Engineering: C 33 (2013) 2550–2554.

[23] C.-K. Chai, A. C. Akyildiz, L. Speelman, F. J. Gijsen, C. W. Oomens, M. R. van Sambeek, A. van der Lugt, F. P. Baaijens, Local axial compressive mechanical properties of human carotid atherosclerotic plaquescharacterisation by indentation test and inverse finite element analysis, Journal of Biomechanics 46 (2013) 1759–1766.

[24] Z. Teng, Y. Zhang, Y. Huang, J. Feng, J. Yuan, Q. Lu, M. P. Sutcliffe, A. J. Brown, Z. Jing, J. H. Gillard, Material properties of components in human carotid atherosclerotic plaques: a uniaxial extension study, Acta Biomaterialia 10 (2014) 5055–5063.

[25] A. V. Kamenskiy, I. I. Pipinos, Y. A. Dzenis, C. S. Lomneth, S. A. J. Kazmi, N. Y. Phillips, J. N. MacTaggart, Passive biaxial mechanical properties and in vivo axial pre-stretch of the diseased human femoropopliteal and tibial arteries, Acta biomaterialia 10 (2014) 1301–1313.

[26] E. M. Cunnane, J. J. Mulvihill, H. E. Barrett, M. T. Walsh, Simulation of human atherosclerotic femoral plaque tissue: the influence of plaque material model on numerical results, Biomedical Engineering Online 14 (2015) S7.

[27] A. Karimi, M. Navidbakhsh, A. Shojaei, A combination of histological analyses and uniaxial tensile tests to determine the material coefficients of the healthy and atherosclerotic human coronary arteries, Tissue and Cell 47 (2015) 152–158.

[28] C.-K. Chai, A. C. Akyildiz, L. Speelman, F. J. Gijsen, C. W. Oomens, M. R. van Sambeek, A. van der Lugt, F. P. Baaijens, Local anisotropic mechanical properties of human carotid atherosclerotic plaques–characterisation by micro-indentation and inverse finite element analysis, Journal of the Mechanical Behavior of Biomedical Materials 43 (2015) 59–68.

[29] M. A. Jankowska, M. Bartkowiak-Jowsa, R. Bedzinski, Experimental and constitutive modeling approaches for a study of biomechanical properties of human coronary arteries, Journal of the Mechanical Behavior of Biomedical Materials 50 (2015) 1–12.

[30] L. A. Davis, S. E. Stewart, C. G. Carsten, B. A. Snyder, M. A. Sutton, S. M. Lessner, Characterization of fracture behavior of human atherosclerotic fibrous caps using a miniature single edge notched tensile test, Acta Biomaterialia 43 (2016) 101–111.

[31] A. Karimi, S. M. Rahmati, T. Sera, S. Kudo, M. Navidbakhsh, A combination of constitutive damage model and artificial neural networks to characterize the mechanical properties of the healthy and atherosclerotic human coronary arteries, Artificial Organs 41 (2017) E103–E117.

[32] M. Kobielarz, M. Kozuń, A. Kuzan, K. Maksymowicz, W. Witkiewicz, C. Pezowicz, The intima with early atherosclerotic lesions is load-bearing component of human thoracic aorta, Biocybernetics and Biomedical Engineering 37 (2017) 35–43.

[33] A. C. Akyildiz, L. Speelman, F. J. Gijsen, Mechanical properties of human atherosclerotic intima tissue, Journal of Biomechanics 47 (2014) 773–783.

[34] A. C. Akyildiz, H. H. G. Hansen, H. A. Nieuwstadt, L. Speelman, C. L. De Korte, A. F. W. van der Steen, F. J. H. Gijsen, A framework for local mechanical characterization of atherosclerotic plaques: combination of ultrasound displacement imaging and inverse finite element analysis, Annals of Biomedical Engineering 44 (2016) 968–979.

[35] S. Avril, M. Bonnet, A.-S. Bretelle, M. Grédiac, F. Hild, P. Ienny, F. Latourte, D. Lemosse, S. Pagano, E. Pagnacco, et al., Overview of identification methods of mechanical parameters based on full-field measurements, Experimental Mechanics 48 (2008) 381.

[36] F. Pierron, M. Grédiac, The virtual fields method: extracting constitutive mechanical parameters from full-field deformation measurements, Springer Science & Business Media, 2012.

[37] F. Pierron, G. Vert, R. Burguete, S. Avril, R. Rotinat, M. Wisnom, Identification of the orthotropic elastic stiffnesses of composites with the virtual fields method: sensitivity study and experimental validation, Strain 43 (2007) 250–259.

[38] H. Chalal, F. Meraghni, F. Pierron, M. Grédiac, Direct identification of the damage behaviour of composite materials using the virtual fields method, Composites Part A: Applied Science and Manufacturing 35 (2004) 841–848.

[39] S. Avril, P. Badel, A. Duprey, Anisotropic and hyperelastic identification of in vitro human arteries from full-field optical measurements, Journal of Biomechanics 43 (2010) 2978–2985.

[40] M. R. Bersi, C. Bellini, P. Di Achille, J. D. Humphrey, K. Genovese, S. Avril, Novel methodology for characterizing regional variations in the material properties of murine aortas, Journal of Biomechanical Engineering 138 (2016) 071005.

[41] G. A. Holzapfel, Nonlinear Solid Mechanics: A Continuum Approach for Engineering, Wiley, 2000.

[42] M. Wong, J. Edelstein, J. Wollman, M. G. Bond, Ultrasonic-pathological comparison of the human arterial wall. verification of intima-media thickness., Arteriosclerosis, Thrombosis, and Vascular Biology 13 (1993) 482–486.

[43] H. Nieuwstadt, S. Fekkes, H. Hansen, C. de Korte, A. van der Lugt, J. Wentzel, A. van der Steen, F. Gijsen, Carotid plaque elasticity estimation using ultrasound elastography, mri, and inverse fea–a numerical feasibility study, Medical Engineering & Physics 37 (2015) 801–807.

[44] C. L. de Korte, H. H. Hansen, A. F. van der Steen, Vascular ultrasound for atherosclerosis imaging, Interface Focus 1 (2011) 565–575.

[45] J. E. Wilhjelm, M.-L. Grønholdt, B. Wiebe, S. K. Jespersen, L. K. Hansen, H. Sillesen, Quantitative analysis of ultrasound b-mode images of carotid atherosclerotic plaque: correlation with visual classification and histological examination, IEEE Transactions on Medical Imaging 17 (1998) 910–922.

[46] H. Hansen, R. Lopata, T. Idzenga, C. L. de Korte, Full 2d displacement vector and strain tensor estimation for superficial tissue using beam-steered ultrasound imaging, Physics in Medicine & Biology 55 (2010) 3201.

[47] I. Berselli, V. Codazzi, Ex vivo mechanical characterization of human atherosclerotic coronary arteries using inflation testing and finite element analysis, Master's thesis, Polytechnic University of Milan, 2018.

[48] S. Kishi, T. A. Magalhaes, R. J. Cerci, M. B. Matheson, A. Vavere, Y. Tanami, P. H. Kitslaar, R. T. George, J. Brinker, J. M. Miller, et al., Total coronary atherosclerotic plaque burden assessment by ct angiography for detecting obstructive coronary artery disease associated with myocardial perfusion abnormalities, Journal of Cardiovascular Computed Tomography 10 (2016) 121–127.

[49] M. Naghavi, P. Libby, E. Falk, S. W. Casscells, S. Litovsky, J. Rumberger, J. J. Badimon, C. Stefanadis, P. Moreno, G. Pasterkamp, et al., From vulnerable plaque to vulnerable patient: a call for new definitions and risk assessment strategies: Part 1, Circulation 108 (2003) 1664–1672.

[50] H. Sillesen, P. Muntendam, A. Adourian, R. Entrekin, M. Garcia, E. Falk, V. Fuster, Carotid plaque burden as a measure of subclinical atherosclerosis: comparison with other tests for subclinical arterial disease in the high risk plaque bioimage study, JACC: Cardiovascular Imaging 5 (2012) 681–689.

[51] D. Xu, D. S. Hippe, H. R. Underhill, M. Oikawa-Wakayama, L. Dong, K. Yamada, C. Yuan, T. S. Hatsukami, Prediction of high-risk plaque development and plaque progression with the carotid atherosclerosis score, JACC: Cardiovascular Imaging 7 (2014) 366–373.

[52] P. R. Moreno, R. A. Lodder, K. R. Purushothaman, W. E. Charash, W. N. Oconnor, J. E. Muller, Detection of lipid pool, thin fibrous cap, and inflammatory cells in human aortic atherosclerotic plaques by near-infrared spectroscopy, Circulation 105 (2002) 923–927.

[53] D. Capecchi, History of Virtual Work Laws: A History of Mechanics Prospective, volume 42, Springer Science & Business Media, 2012.

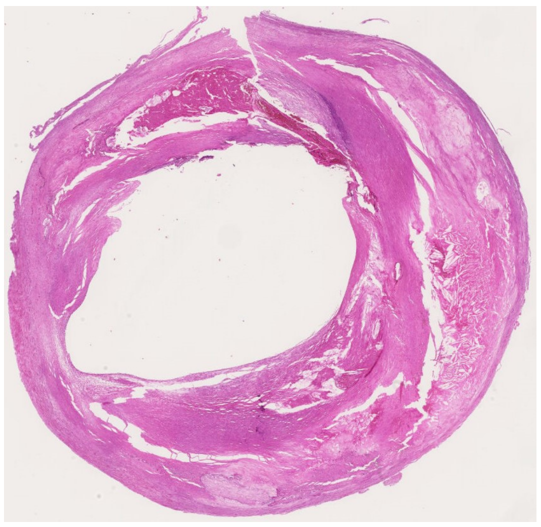
[54] A. Delfino, N. Stergiopulos, J. E. Moore, J. J. Meister, Residual strain effects on the stress field in a thick wall finite element model of the human carotid bifurcation., Journal of Biomechanics 30 (1997) 777–786.

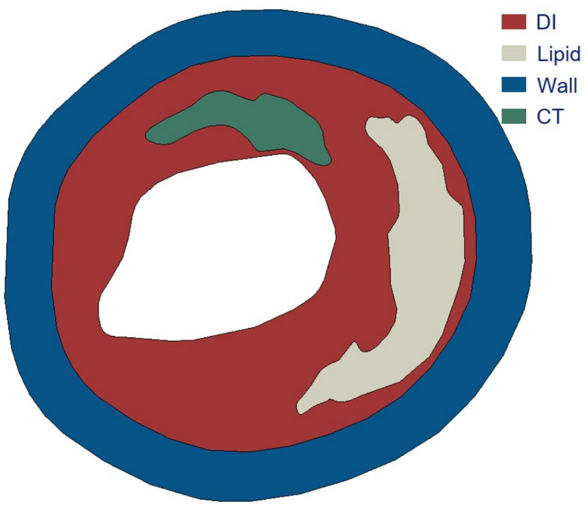
[55] J. D. Humphrey, S. L. O'Rourke, An Introduction to Biomechanics: Solids and Fluids, Analysis and Design, Second Edition, Springer Science & Business Media, 2015.

Appendices

A. Atherosclerosis

Worldwide, cardiovascular diseases such as ischemic heart disease and stroke are ranked as number one and two causes of death [1]. The majority of these cardiovascular clinical events, are mainly triggered by atherosclerosis. Atherosclerosis is a chronic, inflammatory disease of the arterial system [2]. A healthy artery consists of three layers namely, the adventitia, media and intima. In atherosclerotic arteries lipid deposition causes thickening of the intima. Some parts of the atherosclerotic intima can calcify, forming rigid calcified inclusions. Together with the lipid deposition this can result in a complex heterogeneous tissue, referred to as plaque. Atherosclerotic plaque tissue is often categorized in different components namely: the diseased intima containing fibrous tissue, a lipid rich necrotic core, and areas with calcified tissue. Figure 7 shows a cross-section of an atherosclerotic


(a)



(b)

Figure 7: Example of a 2D cross-section of an atherosclerotic human carotid artery, (a) histological slide, (b) segmented into different components, Red=Diseased intima (DI), White=Lipid, Green=Calcified Tissue (CT), Blue=Healthy Wall Tissue.

plaque showing the multiple components present in the plaque tissue. For this study, the adventitia and media were assumed as one healthy wall layer. Current diagnostic techniques are mainly focused on measurement of luminal stenosis [3]. Measurement of luminal stenosis can be a good predictor of cardiovascular events caused by occlusion of the lumen due to intimal thickening. Events can however also occur in patients with mildly stenotic arteries. The main explanation for this is the occurrence of plaque rupture. Rupture of the atherosclerotic plaque can cause thrombus formation. This thrombus can then travel to more distal parts of the body where it can cause a stop in blood flow causing fatal events. For example Rupture of a plaque in the carotid arteries can be the cause of stroke, and rupture of a plaque in one of the coronary arteries can be the cause of an ischemic heart attack.

A lot of studies have been done to asses the risk of plaque rupture. Different studies have introduced vulnerability indices, rupture risk maps and measures of plaque burden [48, 49, 50, 51]. In general, large lipid pools, the presence of inflammatory cells and a thin fibrous cap have been thought to increase the risk of rupture [52]. Here, the fibrous cap is the thin layer of fibrous intima tissue separating a large lipid pool from the lumen. Another approach for predicting rupture is to asses the structural stresses present in the plaque tissue. After all, from a mechanical point of view, rupture occurs when the plaque stress exceeds the plaque strength. To assess plaque stresses computational models such as finite element (FE) models have been used [5, 6, 7, 8, 9]. The results of atherosclerotic plaque FE models greatly depend on the material behavior of the components of the heterogeneous plaque tissue, especially on the one of the diseased intima (DI) component [7].

B. Mechanical characterization of atherosclerotic plaque tissue

Various studies have investigated the material behavior of the atherosclerotic plaque experimentally, using mechanical tests such as uniaxial or biaxial tensile tests and unconfined compression tests [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. For this purpose, the plaque tissue was often cut into smaller parts. An example is given in Figure 8 where the tissue is cut in the shape of a dog bone. The majority of these studies considered the plaque tissue as a homogeneous structure and extracted the composite, gross tissue properties. The reported material stiffness values were shown to span a wide range, even with 2-3 orders of magnitude difference [33]. This great variation was primarily attributed to compositional differences in heterogeneous plaque tissue. Moreover, it has been advocated that more realistic and accurate results would be obtained with testing techniques that mimic the physiological loading environment of the plaque tissue more closely. For atherosclerotic arteries, inflation tests mimic the physiological loading very well. In inflation tests, atherosclerotic arteries are cannulated and pressurised intraluminally (shown in Figure 9). If combined with an appropriate structural imaging technique, they have the potential to provide the heterogeneous material properties of the atherosclerotic plaques through assessing the properties of the mechanically relevant plaque components. A challenge with this approach is that the material properties cannot be assessed straightforward as by applying the standard testing techniques such as tension or compression tests, as mentioned before. Methods such as inverse FE modelling, and VFM are however capable of estimating heterogeneous material stiffness properties from these physiological loading conditions. In this study VFM is used for the material stiffness estimation of atherosclerotic plaque tissue as it is much faster, and therefore shows a greater potential for clinical purpose (section 1).

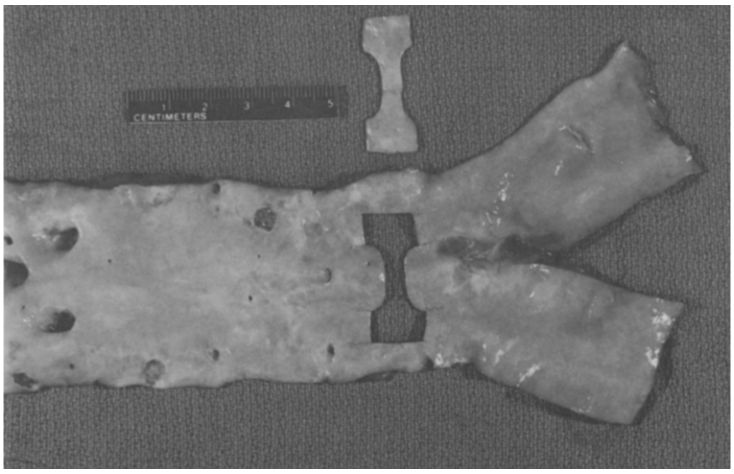


Figure 8: Example of a study where the plaque tissue is cut into a dog bone shape [12].

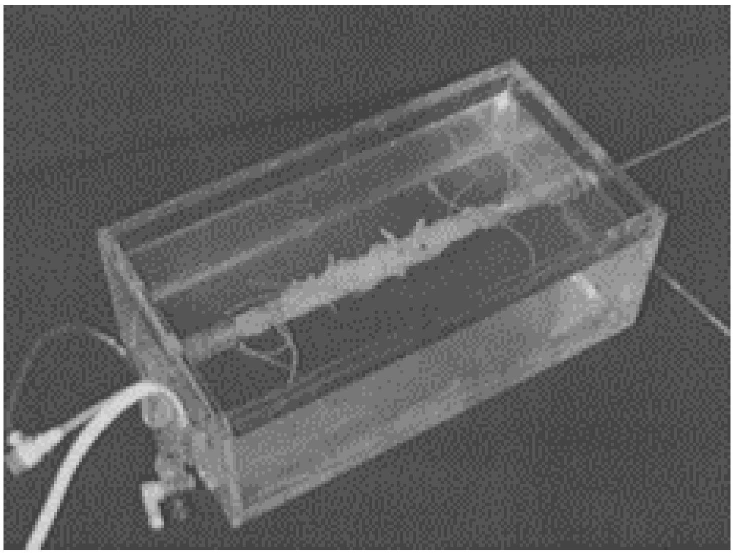


Figure 9: Example of an inflation test setup [14].

C. Virtual Fields Method

VFM is a method used for the estimation of the material stiffness of a structure based on full field deformation data. This section provides a thorough explanation of VFM and the principles of virtual work. For obtaining more insight in the application of VFM a detailed description of a previous study performed on murine aortas using VFM [39] is provided in Appendix H.

VFM is based on the principle of virtual work, which has been used in mechanics for obtaining equilibrium equations since the 17th century [53]. When using a small strain assumption, for a body in static equilibrium free of body forces, the principle of virtual work can be described in a three-dimensional Cartesian coordinate system of x, y, and z in the following form:

$$-\int_V (\sigma_{ij}\delta\varepsilon_{ij})dV + \int_s (t_i\delta u_i)dS = 0, \quad i = x, y, z, \quad j = x, y, z \tag{14}$$

In this equation Einstein notation is used, where a repeated index implies a summation over that index. The internal virtual work (IVW) consists of the integral of the double contraction of the linear stress tensor (σ_{ij}) with the virtual strain field ($\delta\varepsilon_{ij}$) over the entire volume (V) of the body. Whether this volume is defined as the current or reference configuration is not of importance when a small strain assumption is made. The external virtual work (EVW) consists of the integral of the dot product of the traction (t_i) with the virtual displacement field (δu_i) over the surface (S) on which the traction is applied. The EVW then has to be equal to the IVW. Furthermore, first order tensors are denoted with one index (a_i), and second order tensors are denoted with two indices (A_{ij}). The stress and traction from equation 14 represent the real stress present in the body and the real traction that is applied. The virtual displacement field chosen in equation 14, can be any kinematically admissible continuous field. The virtual strain field can then be derived from the virtual displacement field using the small strain definition:

$$\delta\varepsilon_{ij} = \frac{1}{2}(\frac{\partial\delta u_i}{\partial x_j} + \frac{\partial\delta u_j}{\partial x_i}), \quad i = x, y, z, \quad j = x, y, z \tag{15}$$

Here, x_i and x_j are first order tensors, representing the x, y, and z coordinates using a Cartesian coordinate system. For the case of the intraluminally pressurised 2D atherosclerotic artery the small strain definition is not valid. For that reason the principle of virtual work has to be adapted to the principle of virtual work for large deformations. For this study the choice was made to write the principle of virtual work in the current configuration. As shown by Pierron and Grédiac (2012) [36] in their book on the Virtual Fields Method, the principle of virtual work expressed in the current configuration can be written as:

$$-\int_v (\sigma_{ij}\frac{\partial\delta u_i}{\partial x_j})dv + \int_s (t_i\delta u_i)ds = 0, \quad i = x, y, z, \quad j = x, y, z \tag{16}$$

Here, σ_{ij} no longer represents the linear stress tensor but instead represents the Cauchy stress tensor. The internal virtual work is now calculated by integrating over the volume in the current (deformed) configuration. Similarly, the external virtual work is obtained by integrating over the surface on which the traction is applied.

D. Application to 2D atherosclerotic plaque

For the case of the intraluminal pressurization of atherosclerotic plaques, the traction at a certain point on the luminal surface can be defined as:

$$t_i = n_i P \tag{17}$$

Here, P is a scalar value representing the intraluminal pressure, and n_i is a unit vector normal to the luminal surface at the point where the traction vector is calculated.

The material behavior of the plaque tissue was modeled with a Neo-Hookean hyperelastic material model, as described before [23]. The plaque tissue was deemed to be incompressible. The Strain Energy Function (SEF) for this material model is defined as:

$$\Psi = c_1(I_1 - 3) \tag{18}$$

Here, I_1 is the first invariant of the Left Cauchy Green deformation tensor, and c_1 is the material parameter used in the Neo-Hookean material model. The Cauchy stress can then be derived from this SEF by using the definition for Cauchy stress for incompressible materials [41]:

$$\sigma_{ij} = -pI_{ij} + 2\frac{\partial \Psi}{\partial I_1} B_{ij} - 2\frac{\partial \Psi}{\partial I_2} (B_{ij})^{-1} \tag{19}$$

Here, p is the hydrostatic pressure which can be nonuniform within the structure, I_{ij} is the identity tensor, I_1 and I_2 are the first and second invariant of the Left Cauchy Green deformation tensor, and B_{ij} is the left Cauchy Green deformation tensor which is defined as:

$$B_{ij} = \underline{\underline{B}} = \underline{\underline{F}}(\underline{\underline{F}})^T \tag{20}$$

Here, $\underline{\underline{F}}$ is the deformation gradient. Using these definitions, the Cauchy stress for an incompressible Neo-Hookean material model is given as:

$$\sigma_{ij} = -pI_{ij} + 2c_1 B_{ij} \tag{21}$$

Estimation of material parameters for the SEF defining the material behaviour using VFM can be done by choosing the VF such that equation 16 can be easily solved for the material parameters. These VF can be any kinematically admissible field. Although this equation must be valid for any kinematically admissible virtual field, certain choices of VF can help obtaining a solution in an easier way. For this specific application, the choice for an incompressible material model introduces a Lagrange multiplier that is often referred to as the hydrostatic pressure. This hydrostatic pressure enforces the incompressibility in the material. It also introduces an unknown parameter that can be nonuniform within the structure. Because the hydrostatic pressure is different for each material point in the structure and its value is not of interest, VF that cancel out the hydrostatic pressure term are desirable. The hydrostatic pressure term only appears on the diagonal of the stress tensor. This stress tensor is then multiplied with the virtual strain tensor, as shown in equation 16. Making a choice of VF that forces the trace of the virtual strain tensor to be equal to zero will therefore cancel out the hydrostatic pressure. If the hydrostatic pressure term is canceled, it is not necessary to calculate its value for each material point. Using this approach vastly reduces the number of required computations. One other criterion that is important, is that while selecting VF that force the trace of the virtual strain field to be zero, these fields should not cause the integral of the EVW to become zero. If EVW is equal to zero, both terms in equation 16 must be equal to zero. If this would happen, material parameter c_1 cannot be estimated anymore. This results in the following conditions for the VF that have to be satisfied:

$$Tr\left(\frac{\partial \delta u_i}{\partial x_j}\right) = 0 \tag{22}$$

$$\int_s (t_i \delta u_i) ds \neq 0 \tag{23}$$

For this study and specific application, the first virtual displacement field found that met these requirements was:

$$\delta u_i^{[1]} = \begin{bmatrix} \frac{x}{x^2+y^2} \\ \frac{y}{x^2+y^2} \\ 0 \end{bmatrix} \tag{24}$$

Here, x and y are coordinates of a Cartesian coordinate system. One important notion when using these VF, is that a new condition is introduced namely: $x^2 + y^2 \neq 0$. This condition can be satisfied by placing the origin of the coordinate system inside the lumen. This ensures that the VF will not be evaluated at $x^2 + y^2 = 0$ as the intraluminal pressure will cause lumen to expand. The virtual strain tensor is then defined as:

$$\frac{\partial \delta u_i}{\partial x_j} = \begin{bmatrix} \frac{y^2-x^2}{(x^2+y^2)^2} & \frac{-2xy}{(x^2+y^2)^2} & 0 \\ \frac{-2xy}{(x^2+y^2)^2} & \frac{x^2-y^2}{(x^2+y^2)^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{25}$$

Please note that the trace of this virtual strain field is equal to zero which will cause the hydrostatic pressure to cancel out. Therefore the only unknowns left in the equilibrium equation using these VF, are the material parameters used in the SEF to describe the material behaviour. Substituting these VF into equation 16 results in:

$$\begin{aligned} &-\int_v (\sigma_{ij}\frac{\partial \delta u_i}{\partial x_j})dv + \int_s (t_i \delta u_i)ds \\ &= 2c_1 \int_v \left((B_{11} - B_{22}) \cdot \frac{y^2 - x^2}{(x^2+y^2)^2} + 2B_{12} \cdot \frac{-2xy}{(x^2+y^2)^2} \right) dv \\ &\qquad - \int_s \left(\frac{xt_x + yt_y}{x^2+y^2} \right) ds = 0 \end{aligned} \tag{26}$$

Then, discretizing the structure in N elements with finite dimensions, the balance equation (26) becomes:

$$\begin{aligned} &2c_1 \sum_{e=1}^{N} \left[\left((B_{11} - B_{22}) \cdot \frac{y^2 - x^2}{(x^2+y^2)^2} + 2B_{12} \cdot \frac{-2xy}{(x^2+y^2)^2} \right) \cdot A_e \right] \\ &\qquad - \sum_{f=1}^{M} \left[\left(\frac{xt_x + yt_y}{x^2+y^2} \right) \cdot L_f \right] = 0 \end{aligned} \tag{27}$$

Here, N is the total number of grid elements of the discretization and, M is the number of line elements on the luminal surface where the only loading, the intraluminal pressure, is applied. A_e is the area of a surface grid element, L_f is the length of a line element on the luminal surface. B_{11}, B_{22}, B_{12}, x, and y, are evaluated at the center of the grid element. t_x and t_y are the x and y components of the traction vector applied on the luminal surface. Equation 27 describes a homogeneous case of atherosclerotic plaque. Since there is only one single material parameter, c_1, this can be directly obtained from the equation by an analytical solution. Plaque tissue is however heterogeneous. The state-of-the-art approach to incorporating the tissue heterogeneity in biomechanical analyses is segmenting the plaque tissue for the mechanically relevant components, namely lipid pool (L), diseased intima (DI), calcified tissue (CT) and the arterial wall (W), where the individual components are deemed to be homogeneous [7]. Equation 27 can be reorganized such that the internal virtual work is obtained for all tissue components separately by integrating over the volume of these components. In the case of four mechanically distinct plaque components, each modeled as incompressible Neo-Hookean material, the discretized balance equations becomes:

$$\begin{aligned} &c_{DI} \cdot IVW^*_{DI} + c_L \cdot IVW^*_L + c_{CT} \cdot IVW^*_{CT} + c_W \cdot IVW^*_W \\ &\qquad - EVW = 0 \end{aligned} \tag{28}$$

Here, IVW^* of one specific component is defined as:

$$IVW^*_{Component} = 2 \sum_{e=1}^{N^*} \left[\left((B_{11} - B_{22}) \cdot \frac{y^2 - x^2}{(x^2+y^2)^2} + 2B_{12} \cdot \frac{-2xy}{(x^2+y^2)^2} \right) \cdot A_e \right] \tag{29}$$

where N^* is the number of elements inside one specific component.

This approach results in an underdetermined system of one equation with four unknowns (c_1 of each component). Theoretically, four equations would be sufficient to solve this problem. However, due to the presence of noise optimization is used rather than a direct analytical solution. For such a minimization approach, having more equations than unknowns can be beneficial. New VF were obtained through two means. First, a set of new fields were created by multiplying the first virtual field (Equation 24) with the periodic functions of sin and cosine:

$$\delta \underline{u}^{[2n]} = \delta \underline{u}^{[1]} \cdot cos^{2n}(\theta) \tag{30}$$

$$\delta \underline{u}^{[2n+1]} = \delta \underline{u}^{[1]} \cdot sin^{2n}(\theta) \tag{31}$$

Here, $n = 1, 2, 3...$, and θ was defined as the circumference which is related to x and y by:

$$\theta = arccos\left(\frac{x}{\sqrt{x^2+y^2}}\right) = arcsin\left(\frac{y}{\sqrt{x^2+y^2}}\right) \tag{32}$$

This can be rewritten as:

$$cos^2(\theta) = \frac{x^2}{x^2+y^2} \tag{33}$$

and:

$$sin^2(\theta) = \frac{y^2}{x^2+y^2} \tag{34}$$

Then, the gradients of these virtual displacement fields are required. Equations below show the derivations for the virtual strain fields when multiplying the virtual displacement field with periodic functions to the power n.
For:

$$\delta \underline{u}^{[2n]} = \delta \underline{u}^{[1]} \cdot cos^{2n}(\theta) \tag{35}$$

with $n = 1, 2, 3, ...$, the spatial derivatives are defined as:

$$\frac{\partial u_x}{\partial x} = \frac{\left(\frac{x}{\sqrt{x^2+y^2}}\right)^{2n}(2ny^2 - x^2 + y^2)}{(x^2+y^2)^2} \tag{36}$$

$$\frac{\partial u_y}{\partial y} = -\frac{\left(\frac{x}{\sqrt{x^2+y^2}}\right)^{2n}(2ny^2 - x^2 + y^2)}{(x^2+y^2)^2} \tag{37}$$

$$\frac{\partial u_x}{\partial y} = -\frac{xy\left(\frac{x}{\sqrt{x^2+y^2}}\right)^{2n}(2n+2)}{(x^2+y^2)^2} \tag{38}$$

$$\frac{\partial u_y}{\partial x} = \frac{y\left(2ny^2 - 2x^2\right)\left(\frac{x}{\sqrt{x^2+y^2}}\right)^{2n}}{x\left(x^2+y^2\right)^2} \tag{39}$$

For:

$$\delta\underline{u}^{[2n+1]} = \delta\underline{u}^{[1]} \cdot sin^{2n}(\theta) \tag{40}$$

with $n = 1, 2, 3, ...$, the spatial derivatives are defined as:

$$\frac{\partial u_x}{\partial x} = -\frac{\left(\frac{y^2}{x^2+y^2}\right)^{2n/2}(2nx^2 + x^2 - y^2)}{(x^2+y^2)^2} \tag{41}$$

$$\frac{\partial u_y}{\partial y} = \frac{\left(\frac{y^2}{x^2+y^2}\right)^{n}(2nx^2 + x^2 - y^2)}{(x^2+y^2)^2} \tag{42}$$

$$\frac{\partial u_x}{\partial y} = \frac{x\left(2nx^2 - 2y^2\right)\left(\frac{y^2}{x^2+y^2}\right)^{n}}{y\left(x^2+y^2\right)^2} \tag{43}$$

$$\frac{\partial u_y}{\partial x} = -\frac{xy\left(\frac{y^2}{x^2+y^2}\right)^{n}(2n+2)}{(x^2+y^2)^2} \tag{44}$$

A second set of VF were generated by using a counter-clockwise rotation (θ_i) of the original coordinate system defined as:

$$\theta_i = \left(\frac{i}{i_{max}}\right) \cdot \left(180° - \left(\frac{180°}{i_{max}+1}\right)\right) \tag{45}$$

Here, i is an integer number that ranges from 1 to the desired number of rotations (i_{max}). Rotating the coordinate system results in new independent equations. As the VF are symmetric, the maximum rotation angle θ_{max} was always below 180°. This is illustrated in Figure 10, which shows a graphical representation of the virtual strain field $\frac{\partial \delta u_x}{\partial x}$ resulting from multiplying the first virtual displacement field with $sin^8(\theta)$. For performing the rotations, θ_i was used to define the following rotation matrix:

$$\underline{\underline{Q}} = \begin{bmatrix} cos(\theta_i) & sin(\theta_i) & 0 \\ -sin(\theta_i) & cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{46}$$

Rotations were then performed as follows:

$$\underline{\underline{B}}' = (\underline{\underline{Q}})\underline{\underline{B}}(\underline{\underline{Q}})^T \tag{47}$$

$$\underline{x}' = \underline{\underline{Q}}(\underline{x}) \tag{48}$$

Here, $\underline{\underline{B}}'$ is the Left Cauchy Green deformation tensor expressed in the rotated coordinate system, $\underline{\underline{Q}}$ is the rotation tensor, $\underline{x}$' is a first order tensor containing the spatial coordinates expressed in the rotated coordinate system. The rotated Left Cauchy Green tensor was then used in the discretized summation over all the grid elements (Equation 27) to obtain new equations.

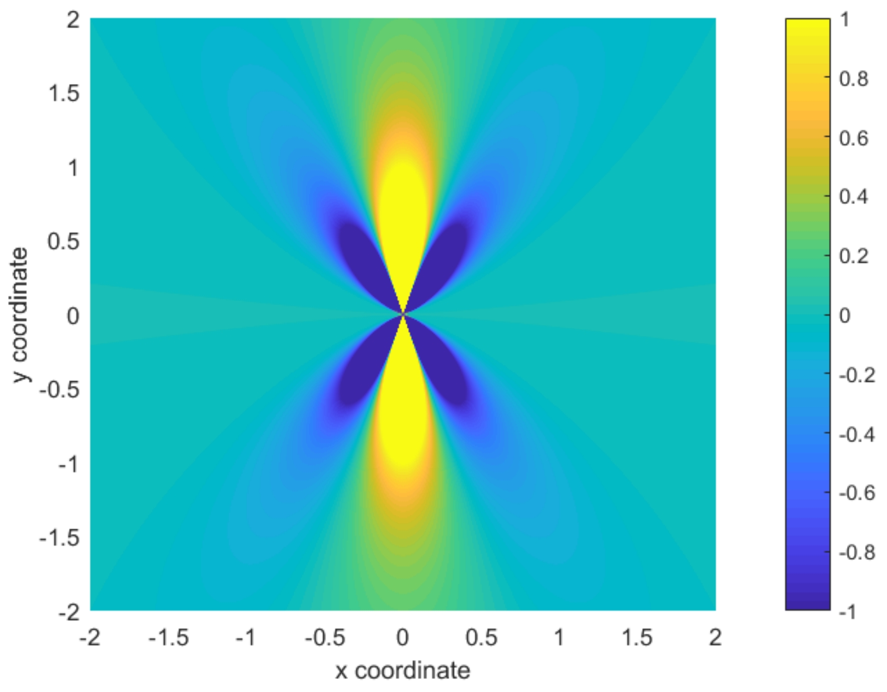


Figure 10: Values for $\frac{\partial \delta u_x}{\partial x}$ when using $\delta \underline{u}^{[9]} = \delta \underline{u}^{[1]} \cdot sin^8(\theta)$, plotted for x and y ranging from -2 to 2 with arbitrary units.

Schematic overview

To further explain the method used in this study, two schematic overviews were made. Figure 11 shows a schematic overview of the different steps involved when using VFM on synthetic data generated by the FE analyses. Figure 12 shows a schematic overview of the different steps involved when using VFM on real data obtained from inflation experiments. One important thing that can be observed from these figures is that the VFM part (indicated in blue) is the same for both situations. The only difference is the way the input of VFM is obtained. The red text in the boxes in both figures indicates the start and end point of the scheme. The MATLAB code that was developed is to be found in Appendix I.

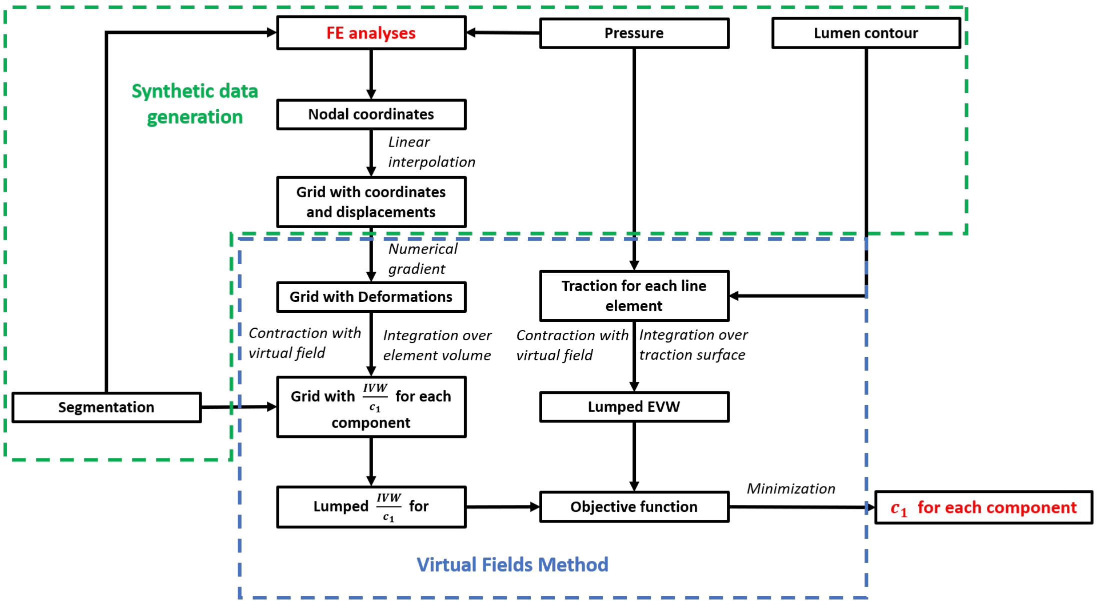

FE analyses
Pressure
Lumen contour
Synthetic data generation
Nodal coordinates
Linear interpolation
Grid with coordinates and displacements
Numerical gradient
Grid with Deformations
Traction for each line element
Contraction with virtual field
Integration over element volume
Contraction with virtual field
Integration over traction surface
Segmentation
Grid with $\frac{IVW}{c_1}$ for each component
Lumped EVW
Lumped $\frac{IVW}{c_1}$ for
Objective function
Minimization
c_1 for each component
Virtual Fields Method

Figure 11: Schematic overview of the different steps involved when using VFM on the synthetic data from FE analyses.

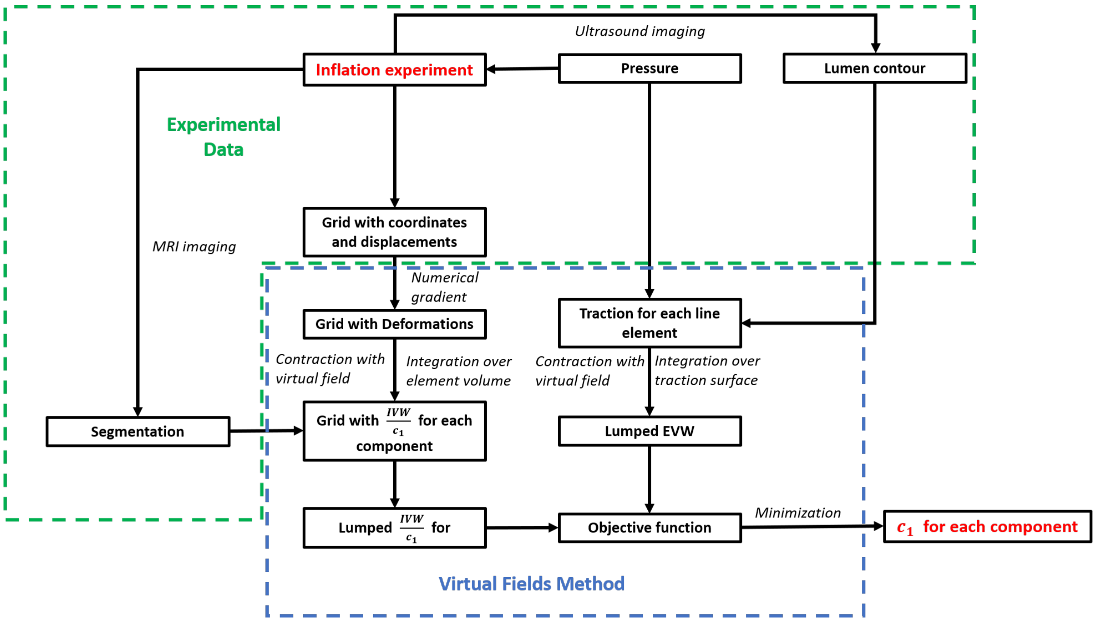

Ultrasound imaging
Inflation experiment
Pressure
Lumen contour
Experimental Data
MRI imaging
Grid with coordinates and displacements
Numerical gradient
Grid with Deformations
Traction for each line element
Contraction with virtual field
Integration over element volume
Contraction with virtual field
Integration over traction surface
Segmentation
Grid with $\frac{IVW}{c_1}$ for each component
Lumped EVW
Lumped $\frac{IVW}{c_1}$ for
Objective function
Minimization
c_1 for each component
Virtual Fields Method

Figure 12: Schematic overview of the different steps involved when using VFM on Real data.

E. Validation

A validation study was performed to evaluate the performance of the developed VFM-based approach for the material characterization of heterogeneous atherosclerotic plaques. This section contains images of all the ten cases used for validation, a more detailed description of methods used for obtaining a grid with displacement data, and a more detailed description of how noise was implemented.

E.1. FE models

Figure 13 and Figure 14 show all the histological slides of human carotid arteries on which the FE models were based. Figure 15 shows the segmentations based on these histological slides.

E.2. Grid generation

As mentioned before, the nodal coordinates in the deformed and undeformed configuration that were generated by the FE analyses were used to generate two grids with two different resolutions. In this section the generation of this grid is explained further. Also, an explanation is given on how the deformation gradient can be calculated using this grid. From the deformation gradient, the Left Cauchy Green deformation tensor can then be calculated which is required for setting up the equations of virtual work for large deformations (Appendix C). FE analyses were used for calculating the nodal displacements caused by the intraluminal pressure. Full details of the FE analyses were given in Section 2.3. Based on the nodal coordinates in the undeformed configuration, a grid was generated with two different resolutions corresponding to the high frequency ultrasound scanner and the clinical scanner. The increment size of this grid was then equal to the resolution of one of the scanners. The preclinical ultrasound scanner was considered to have a resolution of 15 m x 55 m (axial x lateral) [34] and the clinical ultrasound scanner was considered to have a resolution of 75 m x 275 m [46]. Then, linear interpolation was used based on the deformed and undeformed coordinates imported from the FE analyses to create a deformed grid. The numerical gradient was calculated based on the deformed grid and the increment size of the undeformed grid using a central difference algorithm. This numerical gradient was used to calculate the deformation gradient ($\underline{\underline{F}}$) at the center of each grid element. From the deformation gradient, the Left Cauchy Green tensor was calculated using Equation 20. The Left Cauchy Green tensor at the center of each element and the coordinates of each element in the deformed configuration were then used in equation 27.

E.3. Noise evaluation

As there is always noise present in both ex-vivo and in-vivo real-life ultrasound measurements, the influence of noise on the developed VFM-based approach was evaluated as well. In previous studies, the amount of noise present when using a clinical ultrasound scanner was estimated using a phantom experiment [46]. In this study, a vessel phantom with an inner diameter of 3 mm and an outer diameter of 13 mm was intraluminally

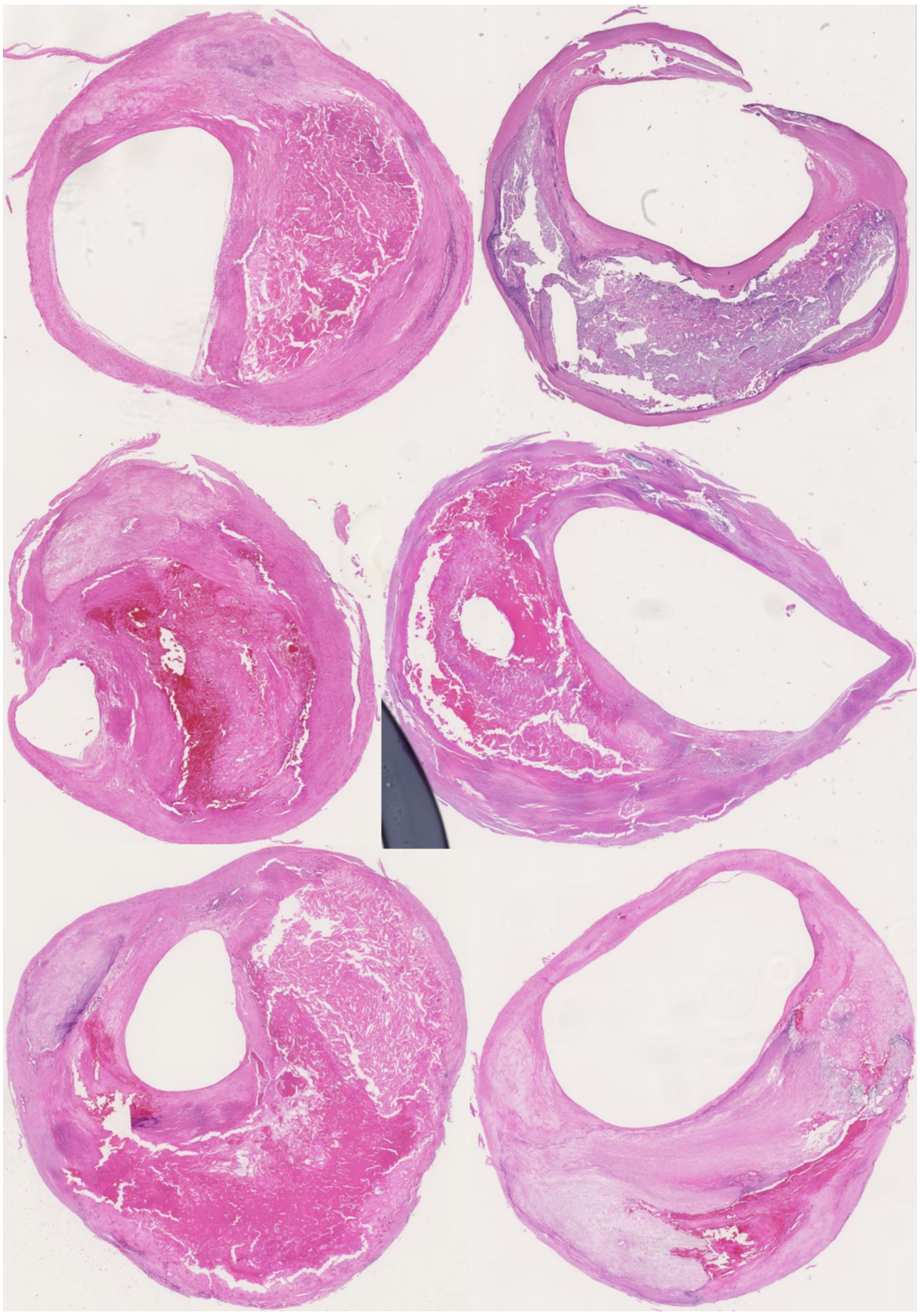


Figure 13: Histological slides of Case 1, 2, 3, 4, 5, and 6.

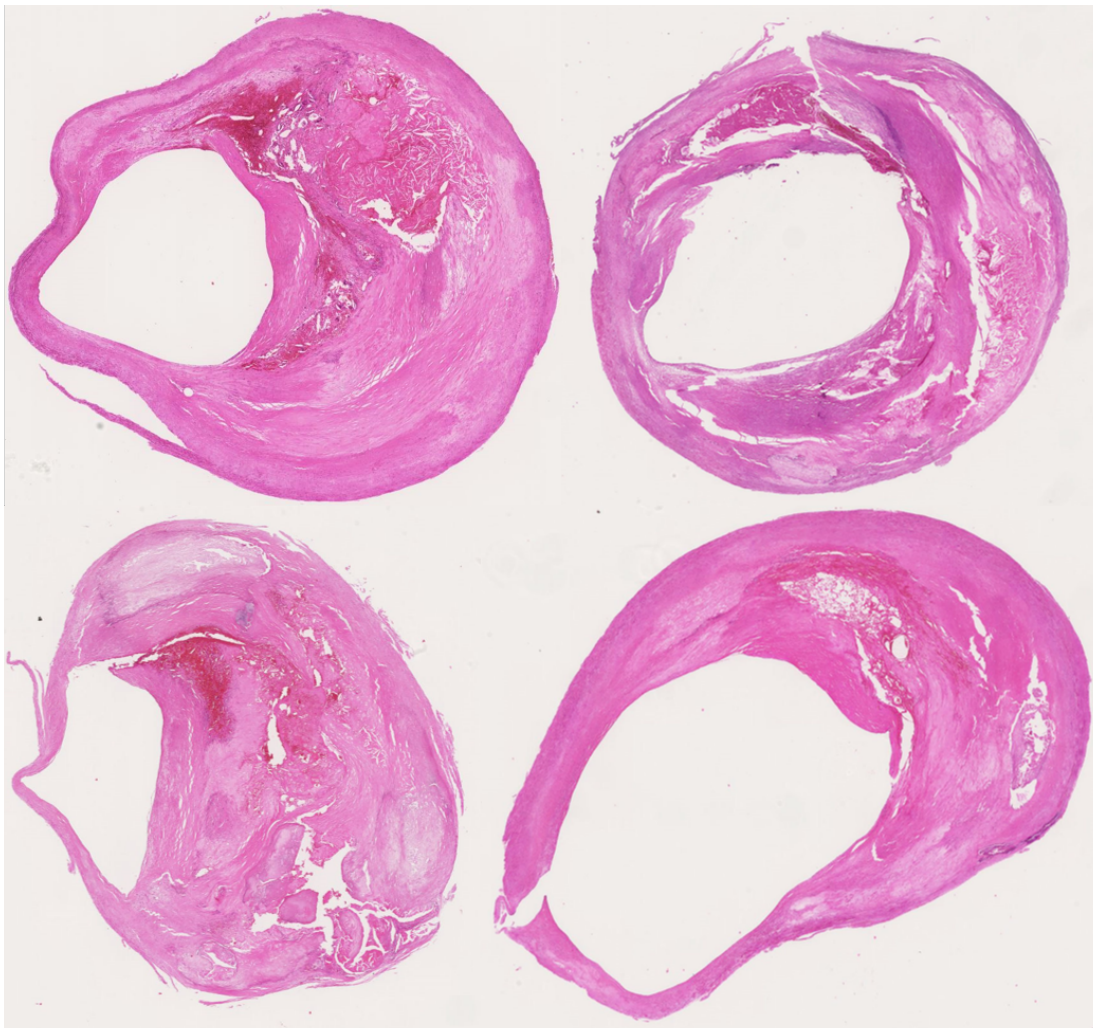
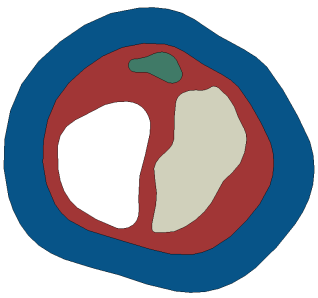
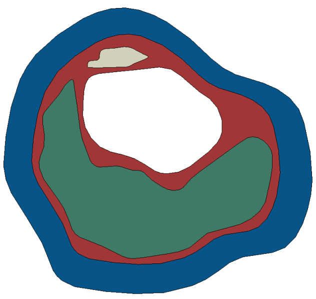


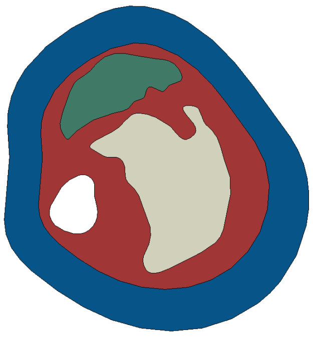
Figure 14: Histological slides of Case 7, 8, 9, and 10.



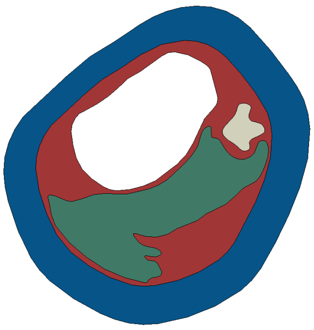
(a)



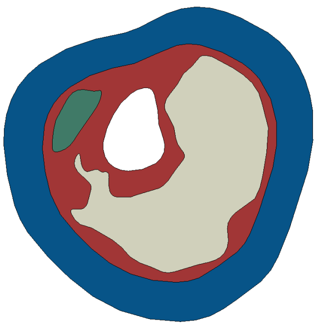
(b)



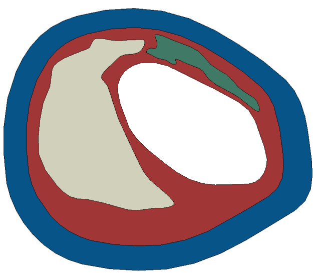
(c)



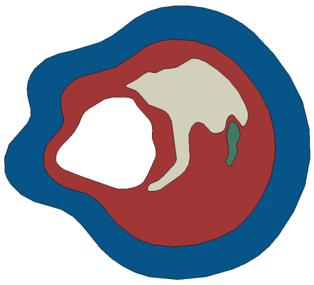
(d)



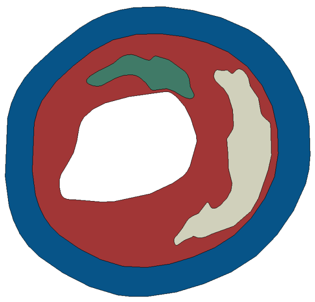
(e)



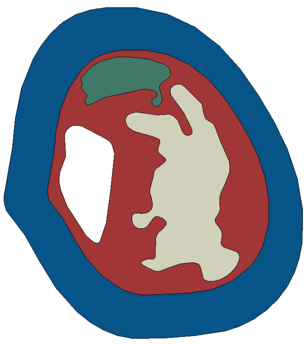
(f)



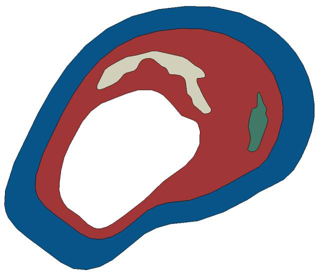
(g)



(h)



(i)



(j)

Figure 15: All of the segmentations based on the histological slides, Red=Diseased intima, White=Lipid, Green=Calcified Tissue, Blue=Healthy Wall Tissue, (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4, (e) Case 5, (f) Case 6, (g) Case 7, (h) Case 8, (i) Case 9, (j) Case 10.

pressurised. The strain measured at the lumen for this study was 1.5%. With a lumen diameter of 3 mm this is equivalent to a displacement of 45 μm at the lumen. The estimated noise in this study was represented by a Root Mean Square Error of 0.63 μm in the axial direction and of 1.8 μm in the lateral direction. Relative to the displacement this is 1.4% in the axial direction and 4.2% in the lateral direction. To not underestimate the noise present in the ultrasound measurements, these numbers were multiplied by a factor two. The expected level of noise at 100 mmHg for the clinical ultrasound scanner was therefore represented by a Root Mean Square Error (RMSE) 3% of the displacement in the axial direction, and 9% of the displacement in the lateral direction [46]. For the high frequency ultrasound scanner, the noise level will scale linearly with the resolution of the scanner [34]. The expected accumulated level of noise at 100 mmHg for the high frequency scanner was therefore represented by a RMSE 0.6% of the displacement in the axial direction, and 1.8% of the displacement in the lateral direction [34, 46]. The c_1-values were recalculated using the noise data.

F. Optimal number of fields

VFM as proposed in this study, can be used to obtain an arbitrary number of equations. This is explained in full detail in Appendix C. An analysis was performed to evaluate the optimal number of the equations (Section 2.4). The relative error in this analysis was calculated as:

$$RE = \left|\frac{c_{VFM} - c_{GT}}{c_{GT}}\right| \tag{49}$$

Here, c_{VFM} is the estimated c_1-value and c_{GT} is the ground truth c_1-value. The following example is given to help the reader understand the analysis that was performed:

1. A grid containing the displacements in either the clinical scanner resolution, or the high frequency scanner resolution is imported into the VFM procedure.
2. A specific number of VF (A) and rotations (B) is used to obtain $A \cdot B$ objective functions.
3. The objective functions are used for minimization using the TRR algorithm, this is done using 100 random initial guesses.
4. From these 100 random initial guesses the mean estimated c_1 value is calculated.
5. The relative error is calculated for this specific number of VF and rotations using Equation 49.
6. The procedure is repeated for a different number of VF and rotations.

G. Results validation

In this section, additional results of the validation study are given that were not reported before.

G.1. Relative error for other components

For this study, the choice was made to use the Relative Error of the diseased intima for finding an optimal combination of VF and rotations. This component was deemed to be the most important plaque component. The other plaque components could also be used for finding an optimal combination of VF and rotations. Figure 16 shows the relative error for all four components for one single case (case 4, shown in Figure 15d). All four

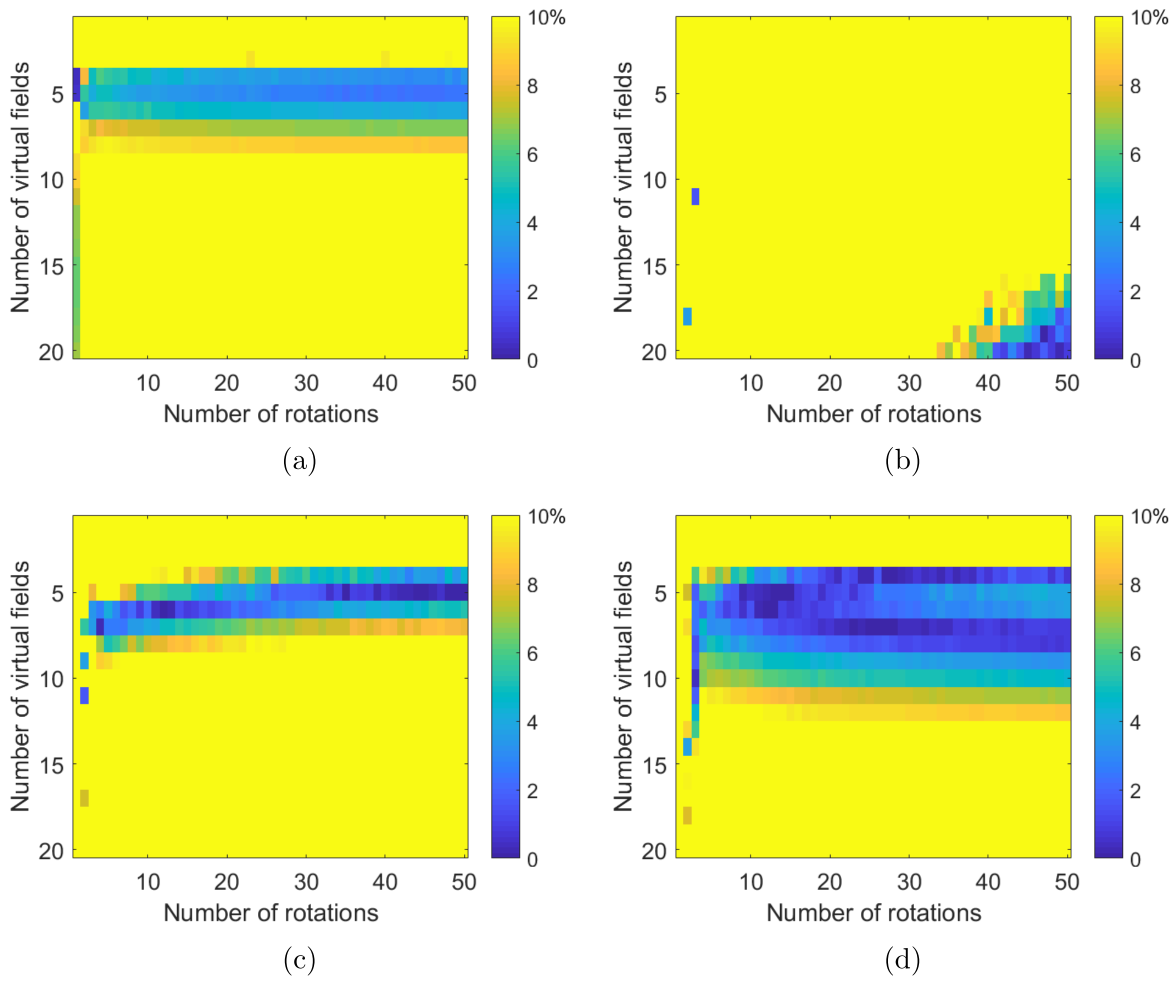
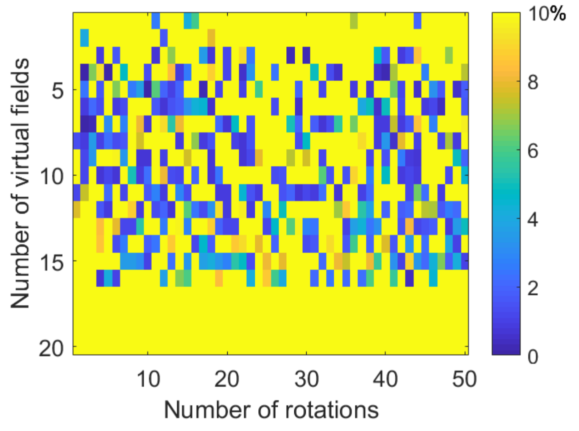


Figure 16: the relative error for all four components for one single case (Figure 15d) (a) diseased intima, (b) lipid, (c) calcified tissue and (d) wall.

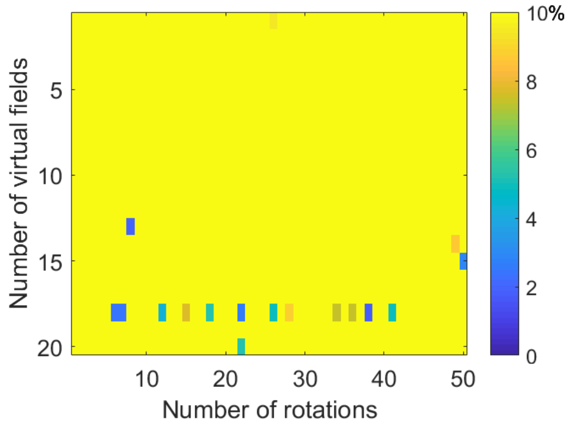
components show a region in which the relative error is low. This region was however not at the same combinations of virtual fields and rotations. The reasons for this are yet unknown, therefore more research is warranted to characterize this phenomenon if other components are deemed more important than the diseased intima.

G.2. Comparison of minimization algorithms

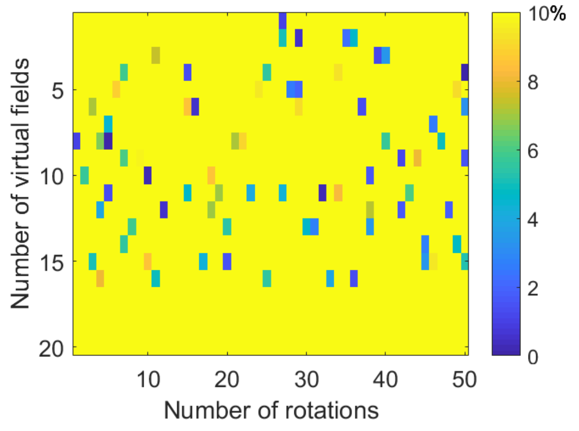
For this study the choice was made to use the 'Trust Region Reflective' minimization algorithm. There are however other minimization algorithms that might perform better or worse. Therefore, in this section the performance of a genetic algorithm is compared to the previously mentioned 'Trust Region Reflective algorithm'. A single objective genetic algorithm was used. This was preferred over a multi-objective genetic algorithm because using a large number of objective functions would result in a large Pareto front. Therefore the absolute values resulting from each objective function were summed to obtain one final objective function. To analyse the differences between the different minimization approaches, the method described in Appendix F for finding an optimal combination of VF and rotations was used. The RE calculated with this method provides a measure for comparing the two minimization algorithms. Figure 17 shows the RE for a single case (case 4, shown in Figure 15d) using the genetic algorithm. Although a lot of combinations of VF and rotations resulted in a RE for the diseased intima that is lower than 10% (Figure 17a), the regions of combinations of VF and rotations resulting in low RE that were observed when using the 'Trust Region Reflective' algorithm were not visible. Instead, the genetic algorithm showed a more random behavior. Due to this random behavior it was difficult to make predictions on which combinations of VF and rotations would result in good estimations of the c_1 material parameters. This random behavior was also observed when calculating the RE using the genetic algorithm for the other cases. For this reason, the genetic algorithm was not used in the final approach.



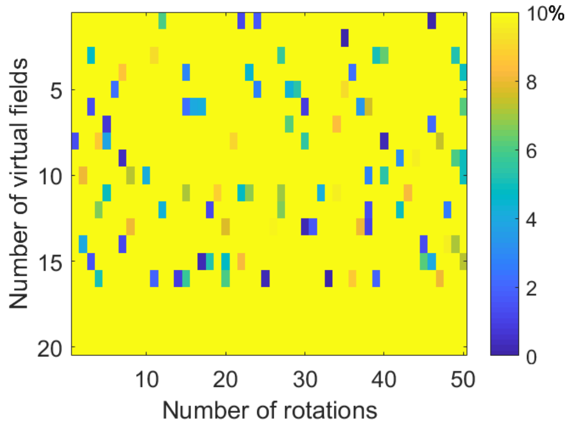
(a)



(b)



(c)



(d)

Figure 17: RE when estimating the c_1 value for the four components using the genetic algorithm, for the geometry shown in Figure 15d, with: (a) diseased intima, (b) lipid, (c) calcified tissue and (d) wall.

H. Virtual Fields Method Avril, 2010

This section contains an explanation of a previous application of VFM. The reader could use this section to understand previous applications of VFM. This could also help understanding the differences between previous approaches and the approach used in the present study. Avril et al. [39] did mechanical characterization of a human aorta using three different nonlinear anisotropic material models. Derivations were not explicitly given in this paper. Here, derivations for one material model proposed by Delfino et al.[54] were performed. The SEF for this model is defined by:

$$\Psi = \frac{\beta \cdot \left(e^{\alpha(I_1-3)} - 1\right)}{2} \tag{50}$$

Here, β, and α are material parameters, and I_1 is the first invariant of the left Cauchy strain tensor. Using Equation 19, the Cauchy stress is defined as:

$$\underline{\underline{\sigma}} = -p\underline{\underline{I}} + \beta\alpha e^{\alpha(I_1-3)}\underline{\underline{B}} \tag{51}$$

This can be rewritten as:

$$\underline{\underline{\sigma}} = \begin{bmatrix} -p + \beta\alpha e^{\alpha(I_1-3)}B_{11} & \beta\alpha e^{\alpha(I_1-3)}B_{12} & \beta\alpha e^{\alpha(I_1-3)}B_{13} \\ \beta\alpha e^{\alpha(I_1-3)}B_{21} & -p + \beta\alpha e^{\alpha(I_1-3)}B_{22} & \beta\alpha e^{\alpha(I_1-3)}B_{23} \\ \beta\alpha e^{\alpha(I_1-3)}B_{31} & \beta\alpha e^{\alpha(I_1-3)}B_{32} & -p + \beta\alpha e^{\alpha(I_1-3)}B_{33} \end{bmatrix} \tag{52}$$

In this paper a local coordinate system was used (shown in Figure 18), defined as:

$$\begin{bmatrix} j \\ k \\ i \end{bmatrix} = \begin{bmatrix} Circumferential \\ Axial \\ Radial \end{bmatrix} \tag{53}$$

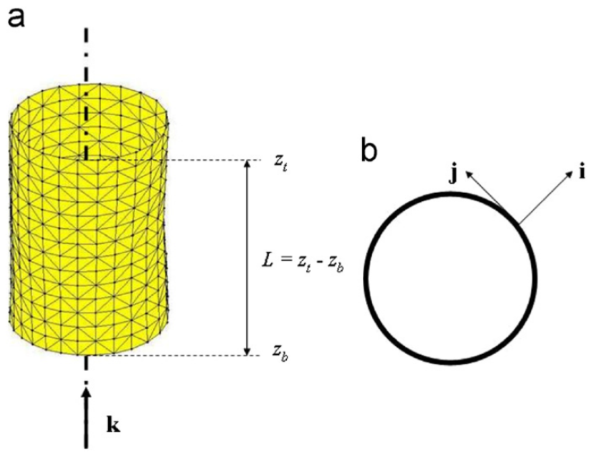


Figure 18: Local coordinate system from the study by Avril et al. [39].

The virtual displacement fields were defined as:

$$\delta \underline{u}_1 = sin\frac{\pi\left(z - z_b\right)}{z_t - z_b} \cdot \underline{\mathbf{i}} \tag{54}$$

and:

$$\delta \underline{u}_2 = \frac{z - z_t}{z_t - z_b} \cdot \underline{\mathbf{k}} \tag{55}$$

Using the coordinate system defined in Equation 53 this leads to:

$$\delta \underline{u}_1 = \begin{bmatrix} 0 \\ 0 \\ sin\frac{\pi(z-z_b)}{z_t-z_b} \end{bmatrix} = \begin{bmatrix} v \\ w \\ u \end{bmatrix} \tag{56}$$

and:

$$\delta \underline{u}_2 = \begin{bmatrix} 0 \\ \frac{z-z_t}{z_t-z_b} \\ 0 \end{bmatrix} = \begin{bmatrix} v \\ w \\ u \end{bmatrix} \tag{57}$$

Here, u, v, and w are the displacements in the radial, circumferential, and axial direction respectively, and z is defined as the component of a coordinate along the longitudinal axis $z = \underline{\mathbf{x}} \cdot \underline{\mathbf{k}}$. Using the coordinate system defined in Equation 53 the virtual strain tensor is defined as:

$$\delta \underline{\underline{\varepsilon}} = \begin{bmatrix} \delta\varepsilon_{\theta\theta} & \delta\varepsilon_{\theta z} & \delta\varepsilon_{\theta r} \\ \delta\varepsilon_{z\theta} & \delta\varepsilon_{zz} & \delta\varepsilon_{zr} \\ \delta\varepsilon_{r\theta} & \delta\varepsilon_{rz} & \delta\varepsilon_{rr} \end{bmatrix} \tag{58}$$

Using a small strain definition, in cylindrical coordinates the strain-displacement relations are defined by[55]:

$$\delta\varepsilon_{rr} = \frac{\partial u}{\partial r} \tag{59}$$

$$\delta\varepsilon_{\theta\theta} = \frac{u}{r} + \frac{1}{r}\frac{\partial v}{\partial \theta} \tag{60}$$

$$\delta\varepsilon_{zz} = \frac{\partial w}{\partial z} \tag{61}$$

$$\delta\varepsilon_{r\theta} = \delta\varepsilon_{\theta r} = \frac{1}{2}\left(\frac{1}{r}\frac{\partial u}{\partial \theta} + \frac{\partial v}{\partial r} - \frac{v}{r}\right) \tag{62}$$

$$\delta\varepsilon_{rz} = \delta\varepsilon_{zr} = \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial r}\right) \tag{63}$$

$$\delta\varepsilon_{\theta z} = \delta\varepsilon_{z\theta} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{1}{r}\frac{\partial w}{\partial \theta}\right) \tag{64}$$

Here, subscripts r, θ, and z, denote the radial, circumferential, and axial direction respectively. For the first virtual displacement field, $\delta\underline{u}_1$, the only nonzero derivative is:

$$\frac{\partial u}{\partial z} = \frac{\pi cos\left(\frac{\pi(z-z_b)}{z_t-z_b}\right)}{z_t - z_b} \tag{65}$$

Also, the first term on the right hand side of equation 60 is nonzero:

$$\frac{u}{r} = \frac{1}{r}sin\left(\frac{\pi\left(z-z_b\right)}{z_t-z_b}\right) \tag{66}$$

Using these definitions, the first virtual strain field is defined as:

$$\delta\underline{\underline{\varepsilon}}_1 = \begin{bmatrix} \frac{1}{r}sin\left(\frac{\pi(z-z_b)}{z_t-z_b}\right) & 0 & 0 \\ 0 & 0 & \frac{1}{2}\frac{\pi cos\left(\frac{\pi(z-z_b)}{z_t-z_b}\right)}{z_t-z_b} \\ 0 & \frac{1}{2}\frac{\pi cos\left(\frac{\pi(z-z_b)}{z_t-z_b}\right)}{z_t-z_b} & 0 \end{bmatrix} \tag{67}$$

For the second virtual field, $\delta\underline{u}_2$, the only nonzero derivative is:

$$\frac{\partial w}{\partial z} = \frac{1}{z_b - z_t} \tag{68}$$

Using these definitions, the second virtual strain field is defined as:

$$\delta\underline{\underline{\varepsilon}}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{z_b-z_t} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{69}$$

The internal virtual work, IVW_n, for the virtual field $\delta\underline{u}_n$ is defined as [39]:

$$IVW_n = e_0 \sum_q \underline{\underline{\sigma}}_q : \delta\underline{\underline{\varepsilon}}_n \cdot A_q \tag{70}$$

Here, summation is performed over all elements. Subscript q is the element number, A_q is the area of an element, and e_0 is the thickness of an element. The thickness of the element is equal for all elements and can therefore be placed outside of the integral. For the first virtual field ($n = 1$), the internal virtual work is defined as:

$$\begin{aligned} IVW_1 = e_0 \sum_q &\left[\left(-p + \beta\alpha e^{\alpha(I_1-3)}B_{11}\right)\left(\frac{1}{r}sin\left(\frac{\pi\left(z-z_b\right)}{z_t-z_b}\right)\right)A_q\right] \\ + e_0 \sum_q &\left[\left(\left(\beta\alpha e^{\alpha(I_1-3)}\right)\left(B_{32}+B_{23}\right)\left(\frac{1}{2}\frac{\pi cos\left(\frac{\pi(z-z_b)}{z_t-z_b}\right)}{z_t-z_b}\right)\right)A_q\right] \end{aligned} \tag{71}$$

For the second virtual field, $(n = 2)$ the internal virtual work is defined as:

$$IVW_2 = e_0 \sum_q \left[\left(-p + \beta\alpha e^{\alpha(I_1-3)} B_{22} \right) \left(\frac{1}{z_b - z_t} \right) A_q \right] \tag{72}$$

In this paper a plane stress condition was used $(\sigma_{33} = 0)$. In this case the third direction is the radial direction. Using the plane stress condition, the Cauchy stress tensor is defined as:

$$\underline{\underline{\sigma}} = \begin{bmatrix} -p + \beta\alpha e^{\alpha(I_1-3)} B_{11} & \beta\alpha e^{\alpha(I_1-3)} B_{12} & 0 \\ \beta\alpha e^{\alpha(I_1-3)} B_{21} & -p + \beta\alpha e^{\alpha(I_1-3)} B_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{73}$$

Substituting this result in Equation 70 gives:

$$IVW_1 = e_0 \sum_q \left[\left(-p + \beta\alpha e^{\alpha(I_1-3)} B_{11} \right) \left(\frac{1}{r} sin \left(\frac{\pi (z - z_b)}{z_t - z_b} \right) \right) A_q \right] \tag{74}$$

The EVW in this study was based on an intraluminal pressure and an axial force. The EVW is defined as [39]:

$$EVW_n = P \sum_q \left[\underline{\mathbf{n}}_q \cdot \delta \underline{u}_n A_q \right] + \delta_{n2} F_n \tag{75}$$

Here, P is the pressure, $\underline{\mathbf{n}}$ is the normal vector on the surface on which the pressure is applied, F is the axial force that is applied, and δ_{n2} is the kronecker delta (for which holds: $\delta_{12} = 0$ and $\delta_{22} = 1$). The EVW for the first virtual field $(n = 1)$ is then defined as:

$$EVW_1 = P \sum_q \left[\underline{\mathbf{n}}_q \cdot \left(sin \frac{\pi (z - zb)}{z_t - z_b} \right) \right] \tag{76}$$

The EVW for the second virtual field $(n = 2)$ is then defined as:

$$EVW_2 = P \sum_q \left[\underline{\mathbf{n}}_q \cdot \left(\frac{z - z_t}{z_b - z_t} \right) \right] + F \tag{77}$$

In this paper, Nelder-Mead minimization was used to obtain parameters β and α. The cost function (ζ) for this minimization is defined as:

$$\zeta = \sum_{n=1}^{2} \left[-IVW_n + EVW_n \right]^2 \tag{78}$$

Using Equations 72, 74 76, and 77, the cost function can be defined as:

$$
\begin{aligned}
\zeta = [e_0 \sum_q \left[\left(-p + \beta\alpha e^{\alpha(I_1-3)} B_{11}\right) \left(\frac{1}{r} sin \left(\frac{\pi\,(z - z_b)}{z_t - z_b}\right)\right) A_q \right] \\
+ P \sum_q \left[\underline{\mathbf{n}}_q \cdot \left(sin \frac{\pi\,(z - z_b)}{z_t - z_b} \right) \right]]^2 \\
+ [-e_0 \sum_q \left[\left(-p + \beta\alpha e^{\alpha(I_1-3)} B_{22}\right) \left(\frac{1}{z_b - z_t}\right) A_q \right] \\
+ P \sum_q \left[\underline{\mathbf{n}}_q \cdot \left(\frac{z - z_t}{z_b - z_t} \right) \right] + F]^2
\end{aligned}
\tag{79}
$$

In this equation the hydrostatic pressure (p) can be obtained from the plane stress condition:

$$
\sigma_{33} = \sigma_{rr} = -p + \beta\alpha e^{\alpha(I_1-3)} B_{33} = 0 \tag{80}
$$

$$
p = \beta\alpha e^{\alpha(I_1-3)} B_{33} \tag{81}
$$

Substituting Equation 81 in Equation 79 results in:

$$
\begin{aligned}
\zeta = [e_0 \sum_q \left[\left(\beta\alpha e^{\alpha(I_1-3)} \left(B_{11} - B_{33}\right)\right) \left(\frac{1}{r} sin \left(\frac{\pi\,(z - z_b)}{z_t - z_b}\right)\right) A_q \right] \\
+ P \sum_q \left[\underline{\mathbf{n}}_q \cdot \left(sin \frac{\pi\,(z - z_b)}{z_t - z_b} \right) \right]]^2 \\
+ [] - e_0 \sum_q \left[\left(\beta\alpha e^{\alpha(I_1-3)} \left(B_{22} - B_{33}\right)\right) \left(\frac{1}{z_b - z_t}\right) A_q \right] \\
+ P \sum_q \left[\underline{\mathbf{n}}_q \cdot \left(\frac{z - z_t}{z_b - z_t} \right) \right] + F]^2
\end{aligned}
\tag{82}
$$

In this section, the derivation is shown for one loading condition. In the study by Avril et al. [39] an inflation test was performed with eight different pressure steps and different axial stretches. The cost function used in the paper for the minimization depends on all loading conditions:

$$
\zeta = \sum_t \sum_{n=1}^{2} \left[-IVW_{tn} + EVW_{tn}\right]^2 \tag{83}
$$

Here the variable t represents the different loading steps. The two variables α, and β, are to be determined in the minimization procedure. Although the approach used in the study by Avril et al. [39] shows similarities to the approach used for the present study, this section shows that there are a lot of differences between the two approaches.

I. MATLAB

I.1. MATLAB code for validation

In this Appendix, the custom-built MATLAB code for the validation study is given. First, the main file is given, then all functions and files that are called upon in the main file are given. Please note that some in between validation steps are now commented.

Main File

Below, the main file is given for estimating the c_1-values based on FE models.

```
clear all
close all
clc

addpath('GEOS_100mmHg')
addpath('Functions')
addpath('GEOS_100_mmHG_inputfiles')
addpath('temp')

DefineParameters_n4

% All other parameters in DefineParameters_n4_i.m

% Define directory to save temporary .mat files]
Directory='temp/';

% Define directory to save grid minimization files
% Directory2=['AllGEO_100/' 'AllGeos_VF' num2str(length(VF_on))
    ...
%       '_R_' num2str(EndNrOfRot)  '/'];
% Directory2=['AllGEO_GOOD_RESO/'];
Directory2=['ALLGEOS_VF7R26/'];

% disp(Directory2) % Create this folder

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO5_100
Filename='GEO5_INP.mat';
save([Directory Filename])

```

```
% Run analysis
MasterFile_GridMini_NoIt2
% Save results
save([Directory2 'GEO5'])
% save('trials/VF6R50_NoNoise')

% save('TryingToFindNiceRepresentative/GEO5_VF9_R50')
%
% xlswrite('TryingToFindNiceRepresentative/GEO5_VF9_R50.xlsx',[
    c_esti_TR_Rv_mean;c_esti_TR_R;c_esti_TR;c_esti_TR_Rv_std])
% writematrix(c_esti_TR_Rv_mean,'Test.xlsx','Sheet',1,'Range','
    E1')
% writetable(c_esti_TR_Rv_mean,'Test.xlsx','Sheet',1,'Range','D1
    ')
%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO6_100
% GEO6
Filename='GEO6_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO6'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO7_100
Filename='GEO7_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO7'])

%% Define Geometry
```

```
clearvars -except Directory2 Directory
DefineParameters_n4

GEO8_100
Filename='GEO8_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO8'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO9_100
Filename='GEO9_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO9'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO10_100
Filename='GEO10_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO10'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

```

```
GEO11_100
Filename='GEO11_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO11'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO12_100
Filename='GEO12_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO12'])
% save([Directory2 'GEO12_5'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO13_100
Filename='GEO13_INP.mat';
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO13'])

%% Define Geometry
clearvars -except Directory2 Directory
DefineParameters_n4

GEO14_100
Filename='GEO14_INP.mat';
```

```
save([Directory Filename])

% Run analysis
MasterFile_GridMini_NoIt
% Save results
save([Directory2 'GEO14'])
```

DefineParameters_n4.m

File defining the parameters used for the VFM approach.

```
% Define ground truth values
GT=zeros(11,4);
GT(:,1)=0.12;
GT(:,2)=0.005;
GT(:,3)=0.423;
GT(:,4)=0.25;
x0=GT(1,:);

% Define increment size, Resolution
% incx=0.275;
% incy=0.075;

% incx=0.055;
% incy=0.015;

incx=0.01;
incy=0.01;

% Define Pressure
Pressure_final=0.01333;
Pressure=0.01333;

% Define Translations if required
translate_x=0;
translate_y=0;
% ROT_a=0;

% Define Noise
% [x noise, y noise]
% noise=(0).*[1.8*10^(-3) 0.63*10^(-3)];
% noise=((16/5)/2).*[1.8*10^(-3) 0.63*10^(-3)];
% noise=(16/2).*[1.8*10^(-3) 0.63*10^(-3)];
noise=[0 0];
```

```

% Define number of VF

imax=4;
% imax=5;
for II=1:imax
    if II<12
        VF_on=[1:II];
%           save('VF_on.mat', 'VF_on')
    elseif II==12
        VF_on=[1:11,16];
%           save('VF_on.mat', 'VF_on')
    elseif II==13
        VF_on=[1:11,16,17];
%           save('VF_on.mat', 'VF_on')
    elseif II>13
        VF_on=[1:11,16,17,18:(II+4)];
    end

end

% Define Number of rotations
% j=0;
% j=1;
% j=5;
% j=6;
% j=12;
% j=100;
% j=6;
j=1;

StartRot=1;
EndNrOfRot= j;
MaxRot=180-(180/(j+1));

MaxRan=100; % number of random initial guesses


% MaxRot=360-(360/(j+1));

NL_GEOM=0; %Always set this to zero, as small strain definition
% has to be used for the virtual strain
```

```
%if NLGEOM==1 % virtual Euler Amansi strain
% elseif NLGEOM==0 % virtual Linear Strain
% elseif NLGEOM==2 % virtual Green Lagrange strain


% Some other parameters
PlotRotFigs=0; % if ==1 plot geometries, if ==0 don't
% ClockIsChecked=0;
AngleEVW=90;
NrOfSteps=10;
StartInc=10;
EndInc=10;
```

GEO5_100.m

File defining where to find certain sections in the input file from the FE model.

```
% Define parameters for specific FE input file
Name_input_file='GEO5_100.inp';
xydef='xydef_GEO5_100_i10';
xydef_1='xydef_GEO5_100_i';

StartRowAllNodes=10;
EndRowAllNodes=17915;
MaxNode=17906;

% Define sections containing nodes of interest
StartRowLumenNodes=36675;
EndRowLumenNodes=36686;
StartRowLipidNodes=36741;
EndRowLipidNodes=36753;
StartRowCalcNodes=36755;
EndRowCalcNodes=36759;
StartRowOutNodes=36688;
EndRowOutNodes=36716;
StartRowOutPlaNodes=36718;
EndRowOutPlaNodes=36739;

% Deformed nodes
Startxydef=20;
Endxydef=17925;
```

MasterFile_GridMini_NoIt2.m

Main file containing the procedure.

```

%% Create synthetic data
load(Filename)
FEM_Data

%% VFM
TimeVar=tic;
VFM
Time=toc(TimeVar);

% DispVars
```

FEM_Data_.m

File generating synthetic data.

```
% This loop is only used if multiple pressure steps
% used, which was not used for this study
Step=1/NrOfSteps;
LENG=StartInc:EndInc;
k=1;
for hh=StartInc:EndInc
    Pressure=Pressure_final*(Step*hh);
    Pressure_vec(k)=Pressure_final*(Step*hh);

    nr=num2str(hh);
    rpt='.rpt';
    xydef=[xydef_1 nr rpt];

    CreateGrid
%     if you want to save other variables, define them in the
    clearvars
%     -excepts in the last step of CreateGrid, This is also
    required if you
%     want to save multiple pressure steps. For example:
%                          B11_MP(k,:,:)=B11;
%     And then define B11_MP in the -except

% Add noise to the grid points
ux=zi1_mid-xi_mid;
uy=zi2_mid-yi_mid;
```

```
if  noise(1)>0 &&  noise(2)>0
% nx=noise(1).*randn(size(ux));
% uxn=ux+nx;
% ny=noise(2).*randn(size(uy));
% uyn=uy+ny;

uxn = ux.*(1 + noise(1).*randn(size(ux)));
uyn = uy.*(1 + noise(2).*randn(size(uy)));


zi1_mid=xi_mid+uxn;
zi2_mid=yi_mid+uyn;
end


     k=k+1;

end
```

CreateGrid.m

File generating grid based on FE nodal coordinates.

```
% CreateGrid

% Load all coordinates of al the nodes in the reference
    configuration
[nodecoordinates]=LoadAllNodesFun(Name_input_file,
    StartRowAllNodes,EndRowAllNodes);

% Create vector containing all nodes
nodes= 1:MaxNode;

% Apply translation on ABAQUS coordinates in MATLAB
% nodecoordinates(:,2)=nodecoordinates(:,2)+translate_x;
% nodecoordinates(:,3)=nodecoordinates(:,3)+translate_y;

% Apply rotation on ABAQUS coordinates in MATLAB
% theta=ROT_a;
% cs = cosd(theta);
% sn = sind(theta);
%
```

```
% New_nodecoordinates(:,2) = nodecoordinates(:,2) .* cs -
    nodecoordinates(:,3) .* sn;
% New_nodecoordinates(:,3) = nodecoordinates(:,2) .* sn +
    nodecoordinates(:,3) .* cs;
% nodecoordinates(:,2)=New_nodecoordinates(:,2);
% nodecoordinates(:,3)=New_nodecoordinates(:,3);


% Find nodecoordinates of interest
[NodeCoordinatesOfInterest, xnode, ynode]=
    FindNodeCoordinatesOfInterest(nodecoordinates, nodes);

% Load all deformed nodes and coordinates
[nodedefcoordinates]=LoadAllDefNodesFun(xydef,Startxydef,
    Endxydef);

% nodedefcoordinates_ori=nodedefcoordinates;
% Apply translation to deformed coordinates
% nodedefcoordinates(:,2)=nodedefcoordinates(:,2)+translate_x;
% nodedefcoordinates(:,3)=nodedefcoordinates(:,3)+translate_y;

% Rotate deformed coordinates ABAQUS coordinates
% New_nodedefcoordinates(:,2) = nodedefcoordinates(:,2) .* cs -
    nodedefcoordinates(:,3) .* sn;
% New_nodedefcoordinates(:,3) = nodedefcoordinates(:,2) .* sn +
    nodedefcoordinates(:,3) .* cs;
% nodedefcoordinates(:,2)=New_nodedefcoordinates(:,2);
% nodedefcoordinates(:,3)=New_nodedefcoordinates(:,3);

%% add noise to ABAQUS coordinates
% ux=nodedefcoordinates(:,2)-nodecoordinates(:,2);
% uy=nodedefcoordinates(:,3)-nodecoordinates(:,3);
%
% nodedefcoordinates_noNoise(:,1)=nodedefcoordinates(:,1);
% nodedefcoordinates_noNoise(:,2)=nodedefcoordinates(:,2);
% nodedefcoordinates_noNoise(:,3)=nodedefcoordinates(:,3);
%
% if noise(1)>0 &&  noise(2)>0
% nx=noise(1).*randn(size(ux));
% uxn=ux+nx;
% ny=noise(2).*randn(size(uy));
% uyn=uy+ny;
```

```
%
% nodedefcoordinates(:,2)=nodecoordinates(:,2)+uxn;
% nodedefcoordinates(:,3)=nodecoordinates(:,3)+uyn;
% end

% Find the coordinates of interest of the deformed nodes and
    coordinates
[NodeCoordinatesOfInterest_def, xnode_def, ynode_def]=
    FindNodeCoordinatesOfInterest(nodedefcoordinates, nodes);

% noise=0;
ang=0;
Coor1_Ref_original=NodeCoordinatesOfInterest(2,:)';
Coor2_Ref_original=NodeCoordinatesOfInterest(3,:)';
Coor1_Final_original=NodeCoordinatesOfInterest_def(2,:)';
Coor2_Final_original=NodeCoordinatesOfInterest_def(3,:)';

[xi,yi,zi1,zi2,zi1_mid,zi2_mid, xi_mid, yi_mid]=
    calc_metric_edit4(ang,Coor1_Ref_original,...
      Coor2_Ref_original,Coor1_Final_original,Coor2_Final_original
         ,incx,incy);

% Find nodes surrounding lumen, lipid and calc, and put them in
    the right
% order.
[nodes_lumen]=FindNodesFun(Name_input_file,StartRowLumenNodes,
    EndRowLumenNodes);
[NoLu, ~, ynode_lu]=FindNodeCoordinatesOfInterest(
    nodedefcoordinates, nodes_lumen);
% NoLu=NoLuInRightOrder_v6(NoLu);
[NoLuUnDef, xnode_luUnDef, ynode_luUnDef]=
    FindNodeCoordinatesOfInterest(nodecoordinates, nodes_lumen);
NoLuUnDef=NoLuInRightOrder_v6(NoLuUnDef);
[NoLu]=NoDefInrightOrder(NoLu, NoLuUnDef);

[nodes_lipid]=FindNodesFun(Name_input_file,StartRowLipidNodes,
    EndRowLipidNodes);
[NoLi, xnode_li, ynode_li]=FindNodeCoordinatesOfInterest(
    nodecoordinates, nodes_lipid);
NoLi=NoLuInRightOrder_v6(NoLi);

```

```
[nodes_calc]=FindNodesFun(Name_input_file,StartRowCalcNodes,
    EndRowCalcNodes);
[NoCa, xnode_ca, ynode_ca]=FindNodeCoordinatesOfInterest(
    nodecoordinates, nodes_calc);
NoCa=NoLuInRightOrder_v6(NoCa);

[nodes_out]=FindNodesFun(Name_input_file,StartRowOutNodes,
    EndRowOutNodes);
[NoOu, xnode_Ou, ynode_Ou]=FindNodeCoordinatesOfInterest(
    nodecoordinates, nodes_out);
NoOu=NoLuInRightOrder_v6(NoOu);

[nodes_outpla]=FindNodesFun(Name_input_file,StartRowOutPlaNodes,
    EndRowOutPlaNodes);
[NoOuPla, ~, ~]=FindNodeCoordinatesOfInterest(nodecoordinates,
    nodes_outpla);
NoOuPla=NoLuInRightOrder_v6(NoOuPla);



%% Create masks
BW_Lu=roipoly(xi_mid,yi_mid,xi_mid,NoLuUnDef(2,:),NoLuUnDef(3,:)
    );
h=find(BW_Lu==1);
q=find(BW_Lu==0);
BW2_Lu=BW_Lu;
BW2_Lu(h)=0;
BW2_Lu(q)=1;
%
% b22ROI_mid=BW2_Lu.*b22_mid;
% b11ROI_mid=BW2_Lu.*b11_mid;
% b12ROI_mid=BW2_Lu.*b12_mid;
% zi1ROI_mid=BW2_Lu.*zi1_mid;
% zi2ROI_mid=BW2_Lu.*zi2_mid;
%
BW_Li=roipoly(xi_mid,yi_mid,xi_mid,NoLi(2,:),NoLi(3,:));
BW_Ca=roipoly(xi_mid,yi_mid,xi_mid,NoCa(2,:),NoCa(3,:));
BW_Ou=roipoly(xi_mid,yi_mid,xi_mid,NoOu(2,:),NoOu(3,:));
BW_OuPla=roipoly(xi_mid,yi_mid,xi_mid,NoOuPla(2,:),NoOuPla(3,:))
    ;


```

```
Area_Li=polyarea(NoLi(2,:),NoLi(3,:));
Area_Ca=polyarea(NoCa(2,:),NoCa(3,:));
Area_Ou=polyarea(NoOu(2,:),NoOu(3,:));
Area_OuPla=polyarea(NoOuPla(2,:),NoOuPla(3,:));
Area_Lu=polyarea(NoLuUnDef(2,:),NoLuUnDef(3,:));

Area_Pla=Area_OuPla-Area_Li-Area_Ca-Area_Lu;
Area_Wall=Area_Ou-Area_OuPla;

% BW_pla=BW2_Lu-BW_Li-BW_Ca;
% BW_pla=BW2_Lu-BW_Li-BW_Ca;
%
% B11=b11ROI_mid;
% B12=b12ROI_mid;
% B22=b22ROI_mid;

clearvars -except B11 B12 B22 BW_pla BW_Ou BW_OuPla BW_Li BW_Ca
    NoLu k  ...
     VF_on Pressure zi1 zi2 zi1_mid zi2_mid x0 StartRot
        EndNrOfRot MaxRot ...
     incx incy BW_Lu ClockIsChecked AngleEVW PlotRotFigs noise xi
        yi ...
     xi_mid yi_mid NLGEOM MaxRan JJ II C_GA C_TR C_TR_R imax
        jmax TIME_V ...
     Filename Directory Directory2 nrOfItsNoise i_noise
        C_TR_R_noise ...
     C_TR_noise C_Ga_noise Area_Ca Area_Li Area_Li Area_Ou
        Area_OuPla ...
     Area_Pla Area_Wall Area_Lu C_TR_R_noise C_TR_R_mean_noise
        C_TR_noise
```

LoadAllNodesFun.m

File loading all undeformed nodes from the ABAQUS input file.

```
function [nodecoordinates]=LoadAllNodesFun(Name_input_file,
    StartRowAllNodes,EndRowAllNodes)
%% Loading input geometry

%% Initialize variables. Outside geometry
filename = Name_input_file;
delimiter = ',';
%The first and the last line of the nodes in the ABAQUS input
    file:
```

```
startRow = StartRowAllNodes;
endRow = EndRowAllNodes;

%% Format string for each line of text:
%   column1: double (%f)
%        column2: double (%f)
%   column3: double (%f)
%        column4: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%f%*s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to
   generate this
% code. If an error occurs for a different file, try
   regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow-startRow+1, '
   Delimiter', delimiter, 'EmptyValue' ,NaN,'HeaderLines',
   startRow-1, 'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so
   no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and
   regenerate the
% script.

%% Create output variable
nodecoordinates = [dataArray{1:end-1}];
%% Clear temporary variables
clearvars filename delimiter startRow endRow formatSpec fileID
   dataArray ans;

end
```

FindNodeCoordinatesOfInterest.m

Function for finding specified nodal coordinates.

```
function [NodeCoordinatesOfInterest, xnode, ynode]=
    FindNodeCoordinatesOfInterest(nodecoordinates, nodes)
% nodes=nodes_lumen;
% nodecoordinates=nodedefcoordinates;
% i=1;
for i=1:length(nodes)
    k=find(nodes(i)==nodecoordinates(:,1));
    xnode(i)=nodecoordinates(k,2);
    ynode(i)=nodecoordinates(k,3);
end

NodeCoordinatesOfInterest = [nodes; xnode; ynode];

end
```

LoadAllDefNodesFun.m

File loading all deformed nodes from the ABAQUS input file.

```
function [nodedefcoordinates]=LoadAllDefNodesFun(xydef,
    Startxydef,Endxydef)
%% Loading input geometry

%% Initialize variables. Outside geometry
filename = xydef;
delimiter = ',';
%The first and the last line of the nodes in the ABAQUS input
    file:
startRow = Startxydef;
endRow = Endxydef;

%% Format string for each line of text:
%   column1: double (%f)
%       column2: double (%f)
%   column3: double (%f)
%       column4: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%f%*s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');
```

```

%% Read columns of data according to format string.
% This call is based on the structure of the file used to
    generate this
% code. If an error occurs for a different file, try
    regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow-startRow+1, '
    Delimiter', delimiter, 'EmptyValue' ,NaN,'HeaderLines',
    startRow-1, 'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so
    no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and
    regenerate the
% script.

%% Create output variable
nodedefcoordinates = [dataArray{1:end-1}];
%% Clear temporary variables
clearvars filename delimiter startRow endRow formatSpec fileID
    dataArray ans;

end
```

calc_metric_edit4.m

File creating grid.

```
function [xi,yi,zi1,zi2,zi1_mid,zi2_mid, xi_mid, yi_mid]=
    calc_metric_edit4(ang,Coor1_Ref_original,...
      Coor2_Ref_original,Coor1_Final_original,Coor2_Final_original
          ,incx,incy)
noise=[0 0];
% noise=(16/5).*[1.8*10^(-3) 0.63*10^(-3)];
RotMatrix = [cosd(ang) -sind(ang);sind(ang) cosd(ang)];
% incx=0.055;
% incy=0.015;
% [xi, yi] = meshgrid(min(COORDCOOR1)-inc:inc:max(COORDCOOR1)+inc
    ,...
%        min(COORDCOOR2)-inc:inc:max(COORDCOOR2)+inc);

Coor1_Ref = RotMatrix (1,1)*Coor1_Ref_original + RotMatrix (1,2)
    *Coor2_Ref_original;
Coor2_Ref = RotMatrix (2,1)*Coor1_Ref_original + RotMatrix (2,2)
    *Coor2_Ref_original;
Coor1_Final = RotMatrix (1,1)*Coor1_Final_original + RotMatrix
    (1,2)*Coor2_Final_original;
Coor2_Final = RotMatrix (2,1)*Coor1_Final_original + RotMatrix
    (2,2)*Coor2_Final_original;


[xi, yi] = meshgrid(min(Coor1_Ref)-incx:incx:max(Coor1_Ref)+incx
    ,...
     min(Coor2_Ref)-incy:incy:max(Coor2_Ref)+incy);

% create mid point griddata coordinates of original griddata
[M,N]=size(xi);
xi_mid = zeros(M-1,N-1);
for i=1:(N-1)
    xi_mid(:,i)=(xi(1,i)+ xi(1,i+1))/2;
end
[M,N]=size(yi);
yi_mid = zeros(M-1,N-1);
for i=1:(M-1)
    yi_mid(i,:)=(yi(i,1)+ yi(i+1,1))/2;
end

% BW_Lu=roipoly(xi_mid,yi_mid,xi_mid,NoLuUnDef(2,:),NoLuUnDef
    (3,:));
% h=find(BW_Lu==1);
% q=find(BW_Lu==0);
% BW2_Lu=BW_Lu;
% BW2_Lu(h)=0;
% BW2_Lu(q)=1;
%
% xi_mid=BW2_Lu.*xi_mid;
% yi_mid=BW2_Lu.*yi_mid;
% create coordinates of nodal points of deformed griddata
```

```
zi1 = griddata(Coor1_Ref,Coor2_Ref,Coor1_Final,xi,yi,'linear');
    %final x coordinate
zi2 = griddata(Coor1_Ref,Coor2_Ref,Coor2_Final,xi,yi,'linear');
    %final y coordinate

% create coordinates of midpoints of deformed griddata
zi1_mid = griddata(Coor1_Ref,Coor2_Ref,Coor1_Final,xi_mid,yi_mid
    ,'linear'); %final x coordinate
zi2_mid = griddata(Coor1_Ref,Coor2_Ref,Coor2_Final,xi_mid,yi_mid
    ,'linear'); %final y coordinate

% % add noise if wanted
% this is an old way of implementing noise, it is
% therefore not used anymore
zi1 = zi1+noise(1).*randn(size(zi1));
zi2 = zi2+noise(2).*randn(size(zi2));

zi1_mid = zi1_mid+noise(1).*randn(size(zi1_mid));
zi2_mid = zi2_mid+noise(2).*randn(size(zi2_mid));

% figure, plot(Coor1_Ref,Coor2_Ref,'.','markersize',12), grid on
%
% h = impoly;
% pos = getPosition(h);
% in = inpolygon(xi,yi,pos(:,1),pos(:,2));
% zi1(in)=NaN;
% zi2(in)=NaN;


% f11 = diff(zi1,1,2)/inc;
% f12 = diff(zi1,1,1)/inc;
% f21 = diff(zi2,1,2)/inc;
% f22 = diff(zi2,1,1)/inc;
% metric_4 = f11(1:end-1,:).*f21(1:end-1,:)+f12(:,1:end-1).*f22
    (:,1:end-1);

% [f11]=gradient(zi1,incx);
% [f12]=gradient(zi1,incy);
% [f22]=gradient(zi2,incy);
% [f21]=gradient(zi2,incx);
% [f11_mid]=gradient(zi1_mid,incx);
% [f12_mid]=gradient(zi1_mid,incy);
% [f22_mid]=gradient(zi2_mid,incy);
% [f21_mid]=gradient(zi2_mid,incx);


% size(metric_2)

end

% figure, surf(metric_1, 'edgecolor','none'),view(2),colorbar,
    title('Metric1');
% figure, surf(metric_2, 'edgecolor','none'),view(2),colorbar,
    title('Metric2');
```

FindNodesFun.m

Finding nodes of interest based on specified sections.

```
function [nodes]=FindNodesFun(Name_input_file,StartRowNodes,
    EndRowNodes)
% Find the nodes of interest

filename = Name_input_file;
fid = fopen(filename,'r');
input = textscan(fid,'%s','delimiter','\n');
input = lower(input{1});     % Using lower() makes the input
    lower-case,
fclose(fid);                 % so that I don't need to worry
    about case-sensitivity

k=1;
for i=StartRowNodes:EndRowNodes %insert lines of n-set with the
    section of interest
string=str2num(input{i});
nodes(k:((k+length(string))-1))=string;
k=k+(length(string));
end

clearvars k input fid filename i string
```

NoLuInRightOrder_v6.m

Reordering the lumen nodes in a counter clockwise manner.

```
function [NoLu_new]=NoLuInRightOrder_v6(NoLu)
% clear all;clc
% load('NoLu.mat')

NoLu_new=zeros(size(NoLu));
NoLu_new(1:3,1)=NoLu(1:3,1);

for i=1:length(NoLu)-1
%       for i=1:307
% i=308;
    if i==1
        x=NoLu(2,:)-NoLu(2,i);
        y=NoLu(3,:)-NoLu(3,i);
%          R=sqrt((x.^2) + (y.^2));
        R=(x.^2) + (y.^2);
        R_sort=sort(R);
        k=find(R==R_sort(2));
        NoLu_new(1:3,i+1)=NoLu(1:3,k(1));
    else
        f=find(NoLu_new(1,i)==NoLu(1,:));
        x=NoLu(2,:)-NoLu(2,f);
        y=NoLu(3,:)-NoLu(3,f);
%          R=sqrt((x.^2) + (y.^2));
        R=(x.^2) + (y.^2);
        R_sort=sort(R);
        k=find(R==R_sort(2));
        h=find(R==R_sort(3));
        g=find(R==R_sort(4));
        j=find(R==R_sort(5));

        if length(k)==1  && isempty(find(NoLu(1,k)==NoLu_new
           (1,:)))==1
                NoLu_new(1:3,i+1)=NoLu(1:3,k);
        elseif length(k)==2 && isempty(find(NoLu(1,k(1))==
           NoLu_new(1,:)))==1
                NoLu_new(1:3,i+1)=NoLu(1:3,k(1));
        elseif length(k)==2 && isempty(find(NoLu(1,k(2))==
           NoLu_new(1,:)))==1
                NoLu_new(1:3,i+1)=NoLu(1:3,k(2));

        elseif  length(h)==1 && isempty(find(NoLu(1,h)==NoLu_new
           (1,:)))==1
                NoLu_new(1:3,i+1)=NoLu(1:3,h);
```

```
            elseif length(h)==2 && isempty(find(NoLu(1,h(1))==
               NoLu_new(1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,h(1));
            elseif length(h)==2 && isempty(find(NoLu(1,h(2))==
               NoLu_new(1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,h(2));

            elseif length(g)==1 && isempty(find(NoLu(1,g)==NoLu_new
               (1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,g);
            elseif length(g)==2 && isempty(find(NoLu(1,g(1))==
               NoLu_new(1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,g(1));
            elseif length(g)==2 && isempty(find(NoLu(1,g(2))==
               NoLu_new(1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,g(2));


            elseif length(j)==1 && isempty(find(NoLu(1,j)==NoLu_new
               (1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,j);
            elseif length(j)==2 && isempty(find(NoLu(1,j(1))==
               NoLu_new(1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,j(1));
            elseif length(j)==2 && isempty(find(NoLu(1,j(2))==
               NoLu_new(1,:)))==1
                    NoLu_new(1:3,i+1)=NoLu(1:3,j(2));


            end


%                if NoLu(1,k(1))== NoLu_new(1,i-1)
%                    NoLu_new(1:3,i+1)=NoLu(1:3,h(2));
%                elseif NoLu(1,h(1))== NoLu_new(1,i-1)
%                    NoLu_new(1:3,i+1)=NoLu(1:3,k(2));
%                elseif NoLu(1,k(2))== NoLu_new(1,i-1)
%                    NoLu_new(1:3,i+1)=NoLu(1:3,h(1));
%                elseif NoLu(1,h(2))== NoLu_new(1,i-1)
%                    NoLu_new(1:3,i+1)=NoLu(1:3,k(1));
%                end
        end
```

```
end
%               A=[608:1211];
%               B=flipud(A');
%               B=[1;B];
%               C=NoLu_new(1,:)'-B;
%               find(C<0);find(C>0)



NoLu_new(1:3,i+2)=NoLu_new(1:3,1);
end
```

NoDefInrightOrder.m

Function setting the deformed coordinates and nodes of the lumen in the same position as the undeformed.

```
function [NoLuNew]=NoDefInrightOrder(NoLu, NoLuUnDef)

for i=1:length(NoLuUnDef)

k=find(NoLu(1,:)==NoLuUnDef(1,i));
NoLuNew(:,i)=NoLu(:,k);

end
```

VFM.m

Main file containing the VFM-based approach.

```
% VFM
% Calculate EVW
[EVW]=CalcEVW(NoLu,Pressure,VF_on,AngleEVW);

% Calculate IVW
[B11,B12,B22]=CalcLCG(zi1_mid,zi2_mid,incx,incy);

%%% Using vectors
[A_ma]=Calc_A_ma(zi1_mid,zi1,zi2);
ii=0;

[IVW_Pla(:,ii+1),IVW_Li(:,ii+1),IVW_Ca(:,ii+1),IVW_Wall(:,ii+1),
    IVW_Homo(:,ii+1) ...
      ,B11_vec,B22_vec,B12_vec ...
      ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec ...
```

```
       ,B11_Li_vec,B22_Li_vec,B12_Li_vec  ...
       ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec  ...
       ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec]  ...
      =Rotations_A(B11,B12,B22,BW_Ca,BW_Li,BW_Ou  ...
       ,BW_OuPla,BW_Lu,zi1_mid,zi2_mid,A_ma,VF_on,PlotRotFigs  ...
       ,NL_GEOM);

if PlotRotFigs==1
hold on;plot(NoLu(2,:),NoLu(3,:),'r');hold off
end

ii=1;
for oo=StartRot:EndNrOfRot
theta=(oo/EndNrOfRot)*MaxRot;
Theta_vec(ii)=theta;


[IVW_Pla(:,ii+1),IVW_Li(:,ii+1),IVW_Ca(:,ii+1),IVW_Wall(:,ii+1),
    IVW_Homo(:,ii+1)]  ...
      =Rotations_B(B11_vec,B22_vec,B12_vec  ...
       ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec  ...
       ,B11_Li_vec,B22_Li_vec,B12_Li_vec  ...
       ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec  ...
       ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec...
       ,zi1_mid,zi2_mid,A_ma,VF_on,theta,PlotRotFigs,NL_GEOM);



[NoLuR]=RotNoLu(NoLu,theta);

if PlotRotFigs==1
hold on;plot(NoLuR(2,:),NoLuR(3,:),'r')
hold off
end

[EVW(:,(ii+1))]=CalcEVW(NoLu,Pressure,VF_on,AngleEVW);



ii=ii+1;
end

```

```

%% Minimization

save('temp/Vars_sum.mat','IVW_Pla','IVW_Li','IVW_Ca','IVW_Wall',
    'EVW' ...
      ,'VF_on')

% Evaluate objective function with GT values
[F] =F_Obj(x0);
[F_ma]=F_Obj_mat(x0);
SumF=sum(abs(F));
% Initial Guess
x_ini=x0;
% x_ini=[1 1 1 1];
%
% Algorithm

%% Trust region reflective
options.Algorithm='trust-region-reflective';
% options.OptimalityTolerance=1.0000e-020;

% options = optimoptions(@lsqnonlin,'Algorithm','trust-region-
    reflective',...
%       'OptimalityTolerance',1.0000e-1060,...
%       'StepTolerance', 1.0000e-1000, ...
% 'MaxFunctionEvaluations',10000, ...
% 'MaxIterations',10000);

[c_esti_TR] = lsqnonlin(@F_Obj,x_ini,[0 0 0 0],[],options);


%% Choose random initial guesses
TimeVar3=tic;
options3.Algorithm='trust-region-reflective';
options3.Display='off';
% MaxRan=100;
c_esti_TR_Rv=zeros(MaxRan,4);
for rrr=1:MaxRan
x_ini2=rand(1,4);
[c_esti_TR_Rv(rrr,:)] = lsqnonlin(@F_Obj,x_ini2,[0 0 0 0],[],
    options3);
end
```

```

RandomPlaqueVec=c_esti_TR_Rv(:,1);
RandomLipidVec=c_esti_TR_Rv(:,2);
RandomCalcVec=c_esti_TR_Rv(:,3);
RandomWallVec=c_esti_TR_Rv(:,4);

c_esti_TR_Rv_mean(1)=mean(RandomPlaqueVec);
c_esti_TR_Rv_mean(2)=mean(RandomLipidVec);
c_esti_TR_Rv_mean(3)=mean(RandomCalcVec);
c_esti_TR_Rv_mean(4)=mean(RandomWallVec);

c_esti_TR_Rv_std(1)=std(RandomPlaqueVec);
c_esti_TR_Rv_std(2)=std(RandomLipidVec);
c_esti_TR_Rv_std(3)=std(RandomCalcVec);
c_esti_TR_Rv_std(4)=std(RandomWallVec);

ER=(c_esti_TR_Rv-x0)./x0;
ER=abs(ER);
[MaxER]=max(ER);
ER_Pla_m=MaxER(1);
ER_Li_m=MaxER(2);
ER_Ca_m=MaxER(3);
ER_Wall_m=MaxER(4);
K=find(ER(:,1)==MaxER(1));
L=find(ER(:,2)==MaxER(2));
M=find(ER(:,3)==MaxER(3));
N=find(ER(:,4)==MaxER(4));

c_esti_TR_R(1)=c_esti_TR_Rv(K(1),1);
c_esti_TR_R(2)= c_esti_TR_Rv(L(1),2);
c_esti_TR_R(3)= c_esti_TR_Rv(M(1),3);
c_esti_TR_R(4)= c_esti_TR_Rv(N(1),4);

TimeRandomIni=toc(TimeVar3);

%% Genetic algorithm USE THIS ONE
% TimeVar2=tic;
% % options2.PopulationSize=200;
% % options2=optimoptions(@ga,'PopulationSize',200,'
    MaxGenerations', Inf,'FunctionTolerance',1E-100);
% % c_esti_ga = ga(@F_Obj2,4,[],[],[],[],[0 0 0 0],[],[],
    options2);
```

```
% c_esti_ga = ga(@F_Obj2,4,[],[],[],[],[0 0 0 0],[],[]);
% TimeGa=toc(TimeVar2);

%% Multiobjective Genetic Algorithm
% TimeVar3=tic;
% optionsmoga.DistanceMeasureFcn = {@distancecrowding,'genotype
    '};
% optionsmoga.ParetoFraction = 0.5;
% FitnessFunction = @F_Obj; % Function handle to the fitness
    function
% numberOfVariables = 4; % Number of decision variables
% lb = [0 0 0 0]; % Lower bound
% ub = []; % Upper bound
% A = []; % No linear inequality constraints
% b = []; % No linear inequality constraints
% Aeq = []; % No linear equality constraints
% beq = []; % No linear equality constraints
% [c_esti_moga,Fval,exitFlag,Output] = gamultiobj(
    FitnessFunction,numberOfVariables,A, ...
%     b,Aeq,beq,lb,ub,optionsmoga);
% TimeMoGa=toc(TimeVar3);

%%
% Some figures for analyses
if PlotRotFigs==1
for tt=1:(1*((EndNrOfRot+1)-(StartRot-1)))
    figure(tt)
    hold on
    axis([-6 6 -6 6])
    hold off
end
end

% figure;plot(F)

% FF=[F(1:length(VF_on));
% F(length(VF_on)+1:2*length(VF_on));
% F(2*length(VF_on)+1:3*length(VF_on))];
%

%%% analyse per angle %%%
%%% here you can plot figures of objective
%%% function
if EndNrOfRot>0
% GG=1;
% FF(GG,:)=F(1:length(VF_on));
% for GG=1:EndNrOfRot
% FF(GG+1,:)=F(((GG*length(VF_on))+1):(GG+1)*length(VF_on));
% end
Theta_vec=[0 Theta_vec];
% figure;plot(Theta_vec,F_ma)
% hold on
% for qq=1:length(VF_on)
%     strng1=num2str(qq);
%     strng2='Virtual Field ';
%     strng3{qq,:}=[strng2 strng1];
%     text(Theta_vec(end), F_ma(qq,end),strng1)
% %     hold on
% % plot(Theta_vec(:),FF(:,qq))
% end
% legend(strng3)
% set(gca, 'XTick', Theta_vec)
% hold off
end

% Calculate homogeneous c value entire structure
c_homo=EVW./IVW_Homo;
% figure;
% tic
% % Genetic algorithm
% x2 = ga(@F_Obj3,4,[],[],[],[],[0 0 0 0],[])
% toc
```

CalcEVW.m

File for calculating the EVW.

```
function [EVW]=CalcEVW(NoLu,Pressure,VF_on,AngleEVW)

external_VP=zeros(length(VF_on),1);
external_VP_i=zeros(length(VF_on),(length(NoLu)-1));

% CheckClockFig
angle=AngleEVW;

for uuu=1:length(VF_on)
```

```
    VF_on_i=VF_on(uuu);

% Calculating
for i = 1:(length(NoLu)-1)
vector = NoLu(2:3,i+1)- NoLu(2:3,i);
[px, py]=rotating(vector(1),vector(2),angle);
normal=[px py]';
lnorm=sqrt((normal(1)^2) + (normal(2)^2));
lvector=sqrt((vector(1)^2)+(vector(2)^2));
% lvector_vector(i)=lvector;
normal=normal/lnorm;
% normalvector(i,:)=normal;
PositionMiddle=NoLu(2:3,i)+ (vector/2);
t=Pressure.*normal;
% t_vector(i,:)=t;
x=PositionMiddle(1);
y=PositionMiddle(2);
Theta=acos(x/sqrt((x^2)+(y^2)));
if VF_on_i==1
    external_VP_i(uuu,i) = lvector*(((x*t(1))+(y*t(2)))/((x^2)+(
        y^2)));
end
if VF_on_i==2
    external_VP_i(uuu,i) = lvector*((((x^3)*t(1))+(((x^2)*y)*t
        (2)))/(((x^2)+(y^2))^2));
end
if VF_on_i==3
   external_VP_i(uuu,i) = lvector*((((x*(y^2))*t(1))+((y^3)*t(2)
       ))/(((x^2)+(y^2))^2));
end
if VF_on_i==4
    external_VP_i(uuu,i) = lvector*((((x^5)*t(1))+(((x^4)*y)*t
        (2)))/(((x^2)+(y^2))^3));
end
if VF_on_i==5
    external_VP_i(uuu,i) = lvector*((((x*(y^4))*t(1))+((y^5)*t
        (2)))/(((x^2)+(y^2))^3));
end
if VF_on_i==6
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^6);
end
if VF_on_i==7
```

```
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^6);
end
if VF_on_i==8
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^8);
end
if VF_on_i==9
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^8);
end
if VF_on_i==10
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^10);
end
if VF_on_i==11
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^10);
end
if VF_on_i==12
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^2).*(
       cos(Theta).^2);
end
if VF_on_i==13
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^4).*(
       cos(Theta).^4);
end
if VF_on_i==14
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^6).*(
       cos(Theta).^6);
end
if VF_on_i==15
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^8).*(
       cos(Theta).^8);
end
if VF_on_i==16
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^12);
end
if VF_on_i==17
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^12);
end
if VF_on_i>17
    iseven = rem(VF_on_i, 2) == 0;
    if iseven ==0
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^(
       VF_on_i-5));
    elseif iseven ==1
```

```
      external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^(
         VF_on_i-4));

      end
end


end


EVW=external_VP;
end
if EVW<0
     EVW=-EVW;
end



end
```

rotating.m

File for rotating 2D vectors.

```
function [px, py]=rotating(x,y,angle)


% x=0;
% y=1;
%
% angle=90;
theta = deg2rad(angle);

cs = cos(theta);
sn = sin(theta);

px = x * cs - y * sn;
py = x * sn + y * cs;
end
```

CalcLCG.m

File for calculating the Left Cauchy Green dormation tensor for each grid point.

```
function [B11,B12,B22]=CalcLCG(zi1_mid,zi2_mid,incx,incy)

[f11_mid,f12_mid]=gradient(zi1_mid,incx,incy);
[f21_mid,f22_mid]=gradient(zi2_mid,incx,incy);
%%

b11_mid = f11_mid.^2+f12_mid.^2;
b12_mid = f11_mid.*f21_mid+f12_mid.*f22_mid;
b22_mid = f21_mid.^2+f22_mid.^2;
B11=b11_mid;
B12=b12_mid;
B22=b22_mid;
% metric_1 = b11-b22;
% metric_2 = b12;
end
```

Calc_A_ma.m

File for calculating the area of the grid elements in the deformed configuration.

```
function [A_ma]=Calc_A_ma(zi1_mid,zi1,zi2)
[M,N]=size(zi1_mid);

A_ma=zeros(M,N);

% for k=1:k
for a=1:M
    for b=1:N
        D_12_x=zi1(a+1,b) - zi1(a,b);
        D_12_y=zi2(a+1,b) - zi2(a,b);
        L_12=sqrt((D_12_x.^2)+(D_12_y.^2));
        D_14_x=zi1(a,b+1) - zi1(a,b);
        D_14_y=zi2(a,b+1) - zi2(a,b);
        L_14=sqrt((D_14_x.^2)+(D_14_y.^2));
        A_e=L_12.*L_14;
        A_ma(a,b)=A_e;

    end
end
```

Rotations_A.m

Main file for calculating IVW^* for the unrotated coordinate system.

```
function [IVW_Pla,IVW_Li,IVW_Ca,IVW_Wall,IVW_Homo ...
      ,B11_vec,B22_vec,B12_vec ...
      ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec ...
      ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
      ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
      ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec] ...
     =Rotations_A(B11,B12,B22,BW_Ca,BW_Li,BW_Ou,BW_OuPla,BW_Lu,
         zi1_mid,zi2_mid,A_ma,VF_on,PlotRotFigs ...
      ,NL_GEOM)
% PreAlloVF16_2

B11_Pla=(BW_OuPla-BW_Lu-BW_Ca-BW_Li).*B11;
B11_Li=BW_Li.*B11;
B11_Ca=BW_Ca.*B11;
B11_Wall=(BW_Ou-BW_OuPla).*B11;
B22_Pla=(BW_OuPla-BW_Lu-BW_Ca-BW_Li).*B22;
B22_Li=BW_Li.*B22;
B22_Ca=BW_Ca.*B22;
B22_Wall=(BW_Ou-BW_OuPla).*B22;
B12_Pla=(BW_OuPla-BW_Lu-BW_Ca-BW_Li).*B12;
B12_Li=BW_Li.*B12;
B12_Ca=BW_Ca.*B12;
B12_Wall=(BW_Ou-BW_OuPla).*B12;

B11_M=(BW_Ou-BW_Lu).*B11;
B22_M=(BW_Ou-BW_Lu).*B22;
B12_M=(BW_Ou-BW_Lu).*B12;



% create vectors, rotate coordinates zi_mid
B11_Pla_vec=B11_Pla(:);
B11_Li_vec=B11_Li(:);
B11_Ca_vec=B11_Ca(:);
B11_Wall_vec=B11_Wall(:);
B22_Pla_vec=B22_Pla(:);
B22_Li_vec=B22_Li(:);
B22_Ca_vec=B22_Ca(:);
B22_Wall_vec=B22_Wall(:);
B12_Pla_vec=B12_Pla(:);
B12_Li_vec=B12_Li(:);
B12_Ca_vec=B12_Ca(:);
```

```
B12_Wall_vec=B12_Wall(:);

B11_vec=B11_M(:);
B22_vec=B22_M(:);
B12_vec=B12_M(:);

A_ma_vec=A_ma(:);

x_vec=zi1_mid(:);
y_vec=zi2_mid(:);

if PlotRotFigs ==1
figure;plot(x_vec,y_vec)
end


for i=1:length(VF_on)
VF_on_i=VF_on(i);
[e11(:,i),e22(:,i),e12(:,i)]=WorkingFieldsVF16_vec(x_vec,y_vec,
    VF_on_i,NLGEOM);
end

[IVW_Ma_Li,IVW_Ma_Ca,IVW_Ma_Pla,IVW_Ma_Wall,IVW_Ma_Homo] ...
     = CalcStress5(B11_vec,B22_vec,B12_vec ...
      ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec ...
      ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
      ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
      ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec ...
      ,e11,e22,e12,A_ma_vec);

[M,N]=size(IVW_Ma_Pla);
for ii=1:N
IVW_Pla(ii,1)=nansum(IVW_Ma_Pla(:,ii));
IVW_Li(ii,1)=nansum(IVW_Ma_Li(:,ii));
IVW_Ca(ii,1)=nansum(IVW_Ma_Ca(:,ii));
IVW_Wall(ii,1)=nansum(IVW_Ma_Wall(:,ii));
IVW_Homo(ii,1)=nansum(IVW_Ma_Homo(:,ii));
end


end
% uitbreiden
```

WorkingFieldsVF16_vec.m

File containing the working virtual fields

```
function [e11,e22,e12]=WorkingFieldsVF16_vec(x_vec,y_vec,VF_on_i
    ,NL_GEOM)
% Mat file containing all working virtual fields
        x=x_vec;
        y=y_vec;

if VF_on_i==1
%%        Old fields
%         e11=((y.^2)-(x.^2))./(((y.^2)+(x.^2)).^2);
%         e22=((x.^2)-(y.^2))./(((y.^2)+(x.^2)).^2);
%         e12=(-2.*x.*y)./(((y.^2)+(x.^2)).^2);

duxdx=-(x.^2 - y.^2)./(x.^2 + y.^2).^2;
duydy=(x.^2 - y.^2)./(x.^2 + y.^2).^2;
duxdy=-(2.*x.*y)./(x.^2 + y.^2).^2;
duydx=-(2.*x.*y)./(x.^2 + y.^2).^2;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end

%        New fields 1 Cos^2

if VF_on_i==2
        duxdx=((3.*(x.^2))./(((x.^2)+(y.^2)).^2))-((4.*(x.^4))
            ./(((x.^2)+(y.^2)).^3));
        duydy=(((x.^2))./(((x.^2)+(y.^2)).^2))-((4.*(x.^2).*(y
            .^2))./(((x.^2)+(y.^2)).^3));
        duxdy=-(4.*(x.^3).*y)./(((x.^2)+(y.^2)).^3);
        duydx=((2.*x.*y)./(((x.^2)+(y.^2)).^2))-(4.*(x.^3).*y)
            ./(((x.^2)+(y.^2)).^3);

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);


end

if VF_on_i==3
```

```
%           New fields 2 sin^2

            duxdx=((y.^2)./(((x.^2)+(y.^2)).^2))-((4.*(x.^2).*(y.^2)
                ).*/((((x.^2)+(y.^2)).^3)));
            duydy=((3.*(y.^2))./(((x.^2)+(y.^2)).^2))-((4.*(y.^4))
                ./((((x.^2)+(y.^2)).^3)));
            duxdy=((2.*x.*y)./(((x.^2)+(y.^2)).^2))-((4.*(x).*(y.^3)
                )./((((x.^2)+(y.^2)).^3)));
            duydx=-((4.*(x).*(y.^3))./((((x.^2)+(y.^2)).^3)));

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end

if VF_on_i==4

%           New fields 3 cos^4

            duxdx=((5.*(x.^4))./(((x.^2)+(y.^2)).^3))-((6.*(x.^6))
                ./(((x.^2)+(y.^2)).^4));
            duydy=(((x.^4))./(((x.^2)+(y.^2)).^3))-((6.*(x.^4).*(y
                .^2))./(((x.^2)+(y.^2)).^4));
            duxdy=-(6.*(x.^5).*y)./(((x.^2)+(y.^2)).^4);
            duydx=((4.*(x.^3).*y)./(((x.^2)+(y.^2)).^3))-(6.*(x.^5)
                .*y)./(((x.^2)+(y.^2)).^4);

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end

if VF_on_i==5

%           New fields 4 sin^4

            duxdx=((y.^4)./(((x.^2)+(y.^2)).^3))-((6.*(x.^2).*(y.^4)
                )./((((x.^2)+(y.^2)).^4)));
            duydy=((5.*(y.^4))./(((x.^2)+(y.^2)).^3))-((6.*(y.^6))
                ./((((x.^2)+(y.^2)).^4)));
            duxdy=((4.*x.*(y.^3))./(((x.^2)+(y.^2)).^3))-((6.*(x).*(
                y.^5))./((((x.^2)+(y.^2)).^4)));
            duydx=-((6.*(x).*(y.^5))./((((x.^2)+(y.^2)).^4)));

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
   NL_GEOM);

end
if VF_on_i==6

%         New fields 5 cos^6

duxdx=(7.*x.^6)./(x.^2 + y.^2).^4 - (8.*x.^8)./(x.^2 + y.^2).^5;
duydy=x.^6./(x.^2 + y.^2).^4 - (8.*x.^6.*y.^2)./(x.^2 + y.^2)
    .^5;
duxdy=-(8.*x.^7.*y)./(x.^2 + y.^2).^5;
duydx=(6.*x.^5.*y)./(x.^2 + y.^2).^4 - (8.*x.^7.*y)./(x.^2 + y
    .^2).^5;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
   NL_GEOM);

end
if VF_on_i==7

          %         New fields 6 sin^6
duxdx=(2.*x.^2.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^2
   - (x.^2./(x.^2 + y.^2) - 1).^3./(x.^2 + y.^2) - (3.*x.*(x
    .^2./(x.^2 + y.^2) - 1).^2.*((2.*x)./(x.^2 + y.^2) - (2.*x
    .^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
duydy=(2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^2
   - (x.^2./(x.^2 + y.^2) - 1).^3./(x.^2 + y.^2) + (6.*x.^2.*y
    .^2.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y.^2).^3;
duxdy=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^2
   + (6.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y.^2)
    .^3;
duydx=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^2
   - (3.*y.*(x.^2./(x.^2 + y.^2) - 1).^2.*((2.*x)./(x.^2 + y.^2)
    - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
   NL_GEOM);

end
```

```
if VF_on_i==8
%           New Fields 7 cos^8
duxdx=(9.*x.^8)./(x.^2 + y.^2).^5 - (10.*x.^10)./(x.^2 + y.^2)
    .^6;
duydy=x.^8./(x.^2 + y.^2).^5 - (10.*x.^8.*y.^2)./(x.^2 + y.^2)
    .^6;
duxdy=-(10.*x.^9.*y)./(x.^2 + y.^2).^6;
duydx=(8.*x.^7.*y)./(x.^2 + y.^2).^5 - (10.*x.^9.*y)./(x.^2 + y
    .^2).^6;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end
if VF_on_i==9

%           New Fields 8 sin^8

duxdx=(x.^2./(x.^2 + y.^2) - 1).^4./(x.^2 + y.^2) - (2.*x.^2.*(x
    .^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^2 + (4.*x.*(x
    .^2./(x.^2 + y.^2) - 1).^3.*((2.*x)./(x.^2 + y.^2) - (2.*x
    .^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
duydy=(x.^2./(x.^2 + y.^2) - 1).^4./(x.^2 + y.^2) - (2.*y.^2.*(x
    .^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^2 - (8.*x.^2.*y
    .^2.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^3;
duxdy=- (2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2)
    .^2 - (8.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y
    .^2).^3;
duydx=(4.*y.*(x.^2./(x.^2 + y.^2) - 1).^3.*((2.*x)./(x.^2 + y
    .^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2) - (2.*x.*
    y.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^2;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end
if VF_on_i==10

%          Virtual fields 9 cos^10


```

```
duxdx=(11.*x.^10)./(x.^2 + y.^2).^6 - (12.*x.^12)./(x.^2 + y.^2).^7;
duydy=x.^10./(x.^2 + y.^2).^6 - (12.*x.^10.*y.^2)./(x.^2 + y.^2).^7;
duxdy=-(12.*x.^11.*y)./(x.^2 + y.^2).^7;
duydx=(10.*x.^9.*y)./(x.^2 + y.^2).^6 - (12.*x.^11.*y)./(x.^2 + y.^2).^7;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);

end

if VF_on_i==11

%           Virtual fields 10 sin^10

duxdx=(2.*x.^2.*(x.^2./(x.^2 + y.^2) - 1).^5)./(x.^2 + y.^2).^2 - (x.^2./(x.^2 + y.^2) - 1).^5./(x.^2 + y.^2) - (5.*x.*(x.^2./(x.^2 + y.^2) - 1).^4.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
duydy=(2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^5)./(x.^2 + y.^2).^2 - (x.^2./(x.^2 + y.^2) - 1).^5./(x.^2 + y.^2) + (10.*x.^2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^3;
duxdy=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^5)./(x.^2 + y.^2).^2 + (10.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^3;
duydx=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^5)./(x.^2 + y.^2).^2 - (5.*y.*(x.^2./(x.^2 + y.^2) - 1).^4.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);


end
if VF_on_i==12
%          Virtual fields 11 sin^2 * cos^2


duxdx=(4.*x.^4.*(x.^2./(x.^2 + y.^2) - 1))./(x.^2 + y.^2).^3 - (3.*x.^2.*(x.^2./(x.^2 + y.^2) - 1))./(x.^2 + y.^2).^2 - (x
    .^3.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))
    ./(x.^2 + y.^2).^2;
duydy=(2.*x.^4.*y.^2)./(x.^2 + y.^2).^4 - (x.^2.*(x.^2./(x.^2 +
    y.^2) - 1))./(x.^2 + y.^2).^2 + (4.*x.^2.*y.^2.*(x.^2./(x.^2
    + y.^2) - 1))./(x.^2 + y.^2).^3;
duxdy=(2.*x.^5.*y)./(x.^2 + y.^2).^4 + (4.*x.^3.*y.*(x.^2./(x.^2
     + y.^2) - 1))./(x.^2 + y.^2).^3;
duydx=(4.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1))./(x.^2 + y.^2).^3
    - (x.^2.*y.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2)
    .^2))./(x.^2 + y.^2).^2 - (2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1)
    )./(x.^2 + y.^2).^2;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end
if VF_on_i==13
%          Virtual fields 12 sin^4 * cos ^4

duxdx=(5.*x.^4.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y.^2).^3
    - (6.*x.^6.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y.^2).^4 +
      (2.*x.^5.*(x.^2./(x.^2 + y.^2) - 1).*((2.*x)./(x.^2 + y.^2)
    - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2).^3;
duydy=(x.^4.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y.^2).^3 -
    (4.*x.^6.*y.^2.*(x.^2./(x.^2 + y.^2) - 1))./(x.^2 + y.^2).^5
    - (6.*x.^4.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y
    .^2).^4;
duxdy=- (4.*x.^7.*y.*(x.^2./(x.^2 + y.^2) - 1))./(x.^2 + y.^2)
    .^5 - (6.*x.^5.*y.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y
    .^2).^4;
duydx=(4.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y.^2)
    .^3 - (6.*x.^5.*y.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y
    .^2).^4 + (2.*x.^4.*y.*(x.^2./(x.^2 + y.^2) - 1).*((2.*x)./(x
    .^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2)
    .^3;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end
if VF_on_i==14

```

```
%           Virtual  fields  13  sin^6 * cos^6

duxdx=(8.*x.^8.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^5
    - (7.*x.^6.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2).^4 -
     (3.*x.^7.*(x.^2./(x.^2 + y.^2) - 1).^2.*((2.*x)./(x.^2 + y
    .^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2).^4;
duydy=(8.*x.^6.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y
    .^2).^5 - (x.^6.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2)
    .^4 + (6.*x.^8.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 +
    y.^2).^6;
duxdy=(8.*x.^7.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2)
    .^5 + (6.*x.^9.*y.*(x.^2./(x.^2 + y.^2) - 1).^2)./(x.^2 + y
    .^2).^6;
duydx=(8.*x.^7.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y.^2)
    .^5 - (6.*x.^5.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 + y
    .^2).^4 - (3.*x.^6.*y.*(x.^2./(x.^2 + y.^2) - 1).^2.*((2.*x)
    ./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y
    .^2).^4;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end
if VF_on_i==15

%           Virtual  fields  14  sin^8 * cos^8

duxdx=(9.*x.^8.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^5
    - (10.*x.^10.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^6
     + (4.*x.^9.*(x.^2./(x.^2 + y.^2) - 1).^3.*((2.*x)./(x.^2 + y
    .^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2).^5;
duydy=(x.^8.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2).^5 -
    (10.*x.^8.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2)
    .^6 - (8.*x.^10.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2 +
     y.^2).^7;
duxdy=- (10.*x.^9.*y.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y
    .^2).^6 - (8.*x.^11.*y.*(x.^2./(x.^2 + y.^2) - 1).^3)./(x.^2
    + y.^2).^7;
duydx=(8.*x.^7.*y.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y.^2)
    .^5 - (10.*x.^9.*y.*(x.^2./(x.^2 + y.^2) - 1).^4)./(x.^2 + y
    .^2).^6 + (4.*x.^8.*y.*(x.^2./(x.^2 + y.^2) - 1).^3.*((2.*x)
    ./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y
```

```matlab
.^2).^5;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);

end
if VF_on_i==16

%            Virtual fields 15 cos^12

duxdx=(13.*x.^12)./(x.^2 + y.^2).^7 - (14.*x.^14)./(x.^2 + y.^2).^8;
duydy=x.^12./(x.^2 + y.^2).^7 - (14.*x.^12.*y.^2)./(x.^2 + y.^2).^8;
duxdy=-(14.*x.^13.*y)./(x.^2 + y.^2).^8;
duydx=(12.*x.^11.*y)./(x.^2 + y.^2).^7 - (14.*x.^13.*y)./(x.^2 + y.^2).^8;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);

end
if VF_on_i==17
%            Virtual fields 16 sin^12

duxdx=(x.^2./(x.^2 + y.^2) - 1).^6./(x.^2 + y.^2) - (2.*x.^2.*(x.^2./(x.^2 + y.^2) - 1).^6)./(x.^2 + y.^2).^2 + (6.*x.*(x.^2./(x.^2 + y.^2) - 1).^5.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
duydy=(x.^2./(x.^2 + y.^2) - 1).^6./(x.^2 + y.^2) - (2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^6)./(x.^2 + y.^2).^2 - (12.*x.^2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^5)./(x.^2 + y.^2).^3;
duxdy=- (2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^6)./(x.^2 + y.^2).^2 - (12.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^5)./(x.^2 + y.^2).^3;
duydx=(6.*y.*(x.^2./(x.^2 + y.^2) - 1).^5.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2) - (2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^6)./(x.^2 + y.^2).^2;


[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);
```

```

end
if VF_on_i==18
%       cos
    duxdx=(15.*x.^14)./(x.^2 + y.^2).^8 - (16.*x.^16)./(x.^2 + y.^2).^9;
    duydy=x.^14./(x.^2 + y.^2).^8 - (16.*x.^14.*y.^2)./(x.^2 + y.^2).^9;
    duxdy=-(16.*x.^15.*y)./(x.^2 + y.^2).^9;
    duydx=(14.*x.^13.*y)./(x.^2 + y.^2).^8 - (16.*x.^15.*y)./(x.^2 + y.^2).^9;

[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);


end
if VF_on_i==19
%    sin
    duxdx=(2.*x.^2.*(x.^2./(x.^2 + y.^2) - 1).^7)./(x.^2 + y.^2).^2 - (x.^2./(x.^2 + y.^2) - 1).^7./(x.^2 + y.^2) - (7.*x.*(x.^2./(x.^2 + y.^2) - 1).^6.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
    duydy=(2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^7)./(x.^2 + y.^2).^2 - (x.^2./(x.^2 + y.^2) - 1).^7./(x.^2 + y.^2) + (14.*x.^2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^6)./(x.^2 + y.^2).^3;
    duxdy=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^7)./(x.^2 + y.^2).^2 + (14.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^6)./(x.^2 + y.^2).^3;
    duydx=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^7)./(x.^2 + y.^2).^2 - (7.*y.*(x.^2./(x.^2 + y.^2) - 1).^6.*((2.*x)./(x.^2 + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);


[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,NL_GEOM);


end
if VF_on_i==20
% cos
```

```
        duxdx=(17.*x.^16)./(x.^2 + y.^2).^9 - (18.*x.^18)./(x.^2 + y
            .^2).^10;
        duydy=x.^16./(x.^2 + y.^2).^9 - (18.*x.^16.*y.^2)./(x.^2 + y
            .^2).^10;
        duxdy=-(18.*x.^17.*y)./(x.^2 + y.^2).^10;
        duydx=(16.*x.^15.*y)./(x.^2 + y.^2).^9 - (18.*x.^17.*y)./(x
            .^2 + y.^2).^10;



 [e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
     NL_GEOM);

 end
 if VF_on_i==21
%     sin
        duxdx=(x.^2./(x.^2 + y.^2) - 1).^8./(x.^2 + y.^2) - (2.*x
            .^2.*(x.^2./(x.^2 + y.^2) - 1).^8)./(x.^2 + y.^2).^2 +
            (8.*x.*(x.^2./(x.^2 + y.^2) - 1).^7.*((2.*x)./(x.^2 + y
            .^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
        duydy=(x.^2./(x.^2 + y.^2) - 1).^8./(x.^2 + y.^2) - (2.*y
            .^2.*(x.^2./(x.^2 + y.^2) - 1).^8)./(x.^2 + y.^2).^2 -
            (16.*x.^2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^7)./(x.^2 + y
            .^2).^3;
        duxdy=- (2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^8)./(x.^2 + y
            .^2).^2 - (16.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^7)./(x
            .^2 + y.^2).^3;
        duydx=(8.*y.*(x.^2./(x.^2 + y.^2) - 1).^7.*((2.*x)./(x.^2 +
           y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2) -
            (2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^8)./(x.^2 + y.^2)
            .^2;


 [e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
     NL_GEOM);


 end

 if VF_on_i==22
% cos
```

```
    duxdx=(19.*x.^18)./(x.^2 + y.^2).^10 - (20.*x.^20)./(x.^2 +
        y.^2).^11;
    duydy=x.^18./(x.^2 + y.^2).^10 - (20.*x.^18.*y.^2)./(x.^2 +
        y.^2).^11;
    duxdy=-(20.*x.^19.*y)./(x.^2 + y.^2).^11;
    duydx=(18.*x.^17.*y)./(x.^2 + y.^2).^10 - (20.*x.^19.*y)./(x
        .^2 + y.^2).^11;


[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

end
if VF_on_i==23
% sin
    duxdx=(2.*x.^2.*(x.^2./(x.^2 + y.^2) - 1).^9)./(x.^2 + y.^2)
        .^2 - (x.^2./(x.^2 + y.^2) - 1).^9./(x.^2 + y.^2) - (9.*x
        .*(x.^2./(x.^2 + y.^2) - 1).^8.*((2.*x)./(x.^2 + y.^2) -
        (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);
    duydy=(2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^9)./(x.^2 + y.^2)
        .^2 - (x.^2./(x.^2 + y.^2) - 1).^9./(x.^2 + y.^2) + (18.*
        x.^2.*y.^2.*(x.^2./(x.^2 + y.^2) - 1).^8)./(x.^2 + y.^2)
        .^3;
    duxdy=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^9)./(x.^2 + y.^2)
        .^2 + (18.*x.^3.*y.*(x.^2./(x.^2 + y.^2) - 1).^8)./(x.^2
        + y.^2).^3;
    duydx=(2.*x.*y.*(x.^2./(x.^2 + y.^2) - 1).^9)./(x.^2 + y.^2)
        .^2 - (9.*y.*(x.^2./(x.^2 + y.^2) - 1).^8.*((2.*x)./(x.^2
         + y.^2) - (2.*x.^3)./(x.^2 + y.^2).^2))./(x.^2 + y.^2);


[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);


end
if VF_on_i>23
    iseven = rem(VF_on_i, 2) == 0;
    if iseven ==0
%           sin
% % %       ^12 --> VF_on_i==17
% duxdx17_N = -(y.^12.*(13.*x.^2 - y.^2))./(x.^2 + y.^2).^8
% % %      ^14 —> VF_on_i==19
% duxdx19_N = -(y.^14.*(15.*x.^2 - y.^2))./(x.^2 + y.^2).^9
% % %      ^16 —> VF_on_i==21
% duxdx21_N = -(y.^16.*(17.*x.^2 - y.^2))./(x.^2 + y.^2).^10
% % %      ^18 —> VF_on_i==23
% duxdx23_N = -(y.^18.*(19.*x.^2 - y.^2))./(x.^2 + y.^2).^11
%
%
% % %  Iterative form:
duxdx= -(y.^(VF_on_i-5).*((VF_on_i-4).*x.^2 - y.^2))./(x.^2 + y
    .^2).^((VF_on_i-1)/2);



% duydy
% % %      ^14 —> VF_on_i==19
% duydy19_N = (y.^14.*(15.*x.^2 - y.^2))./(x.^2 + y.^2).^9
% % %      ^16 —> VF_on_i==21
% duydy21_N = (y.^16.*(17.*x.^2 - y.^2))./(x.^2 + y.^2).^10
% % %      ^18 —> VF_on_i==23
% duydy23_N = (y.^18.*(19.*x.^2 - y.^2))./(x.^2 + y.^2).^11

% % %  Iterative form:
duydy = (y.^(VF_on_i-5).*((VF_on_i-4).*x.^2 - y.^2))./(x.^2 + y
    .^2).^((VF_on_i-1)/2);


% duxdy
% % %      ^14 —> VF_on_i==19
% duxdy19_N =(2.*x.*y^13.*(7.*x^2 - y^2))./(x^2 + y^2)^9

% % %      ^16 —> VF_on_i==21
% duxdy21_N =(2.*x.*y^15.*(8.*x^2 - y^2))./(x^2 + y^2)^10

% % %      ^18 —> VF_on_i==23
% duxdy23_N = (2.*x.*y^17.*(9.*x^2 - y^2))./(x^2 + y^2)^11

% % %  Iterative form:
duxdy = (2.*x.*y.^(VF_on_i-6).*((((VF_on_i-1)/2)-2).*x.^2 - y
    .^2))./(x.^2 + y.^2).^((VF_on_i-1)/2);


```

```
% duydx
% % %      ^14 —> VF_on_i==19
% duydx19_N = −(16.*x.*y.^15)./(x.^2 + y.^2).^9
%
% % %      ^16 —> VF_on_i==21
% duydx21_N = −(18.*x.*y.^17)./(x.^2 + y.^2).^10
%
% % %      ^18 —> VF_on_i==23
% duydx23_N = −(20.*x.*y.^19)./(x.^2 + y.^2).^11
%

% % %  Iterative form:
duydx = −((VF_on_i −3).*x.*y.^(VF_on_i −4))./(x.^2 + y.^2).^((
    VF_on_i −1)/2);


[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

      elseif iseven ==1
%          cos

% % %      ^14 —> VF_on_i==18
% duxdx=(15.*x.^14)./(x.^2 + y.^2).^8 − (16.*x.^16)./(x.^2 + y
    .^2).^9;
% % %      ^16 —> VF_on_i==20
% duxdx=(17.*x.^16)./(x.^2 + y.^2).^9 − (18.*x.^18)./(x.^2 + y
    .^2).^10;
% % %      ^18 —> VF_on_i==22
% duxdx=(19.*x.^18)./(x.^2 + y.^2).^10 − (20.*x.^20)./(x.^2 + y
    .^2).^11;
% % %  Iterative form:
duxdx=((VF_on_i −3).*x.^(VF_on_i −4))./(x.^2 + y.^2).^((VF_on_i/2)
    −1) − ((VF_on_i −2).*x.^(VF_on_i −2))./(x.^2 + y.^2).^(VF_on_i
    /2);

% % %      ^14 —> VF_on_i==18
% duydy=x.^14./(x.^2 + y.^2).^8 − (16.*x.^14.*y.^2)./(x.^2 + y
    .^2).^9;
% % %      ^16 —> VF_on_i==20
% duydy=x.^16./(x.^2 + y.^2).^9 − (18.*x.^16.*y.^2)./(x.^2 + y
    .^2).^10;
```

```
% % %      ^18 —> VF_on_i==22
% duydy=x.^18./(x.^2 + y.^2).^10 - (20.*x.^18.*y.^2)./(x.^2 + y
    .^2).^11;
% % %  Iterative form:
duydy=x.^(VF_on_i-4)./(x.^2 + y.^2).^((VF_on_i/2)-1) - ((VF_on_i
    -2).*x.^(VF_on_i-4).*y.^2)./(x.^2 + y.^2).^(VF_on_i/2);


% % %      ^14 —> VF_on_i==18
%      duxdy=-(16.*x.^15.*y)./(x.^2 + y.^2).^9;
% % %      ^16 —> VF_on_i==20
%      duxdy=-(18.*x.^17.*y)./(x.^2 + y.^2).^10;
% % %      ^18 —> VF_on_i==22
%      duxdy=-(20.*x.^19.*y)./(x.^2 + y.^2).^11;
% % %  Iterative form:
duxdy=-((VF_on_i-2).*x.^(VF_on_i-3).*y)./(x.^2 + y.^2).^(VF_on_i
    /2);

% % %      ^14 —> VF_on_i==18
%      duydx=(14.*x.^13.*y)./(x.^2 + y.^2).^8 - (16.*x.^15.*y)./(
    x.^2 + y.^2).^9;
% % %      ^16 —> VF_on_i==20
%      duydx=(16.*x.^15.*y)./(x.^2 + y.^2).^9 - (18.*x.^17.*y)./(
    x.^2 + y.^2).^10;
% % %      ^18 —> VF_on_i==22
%      duydx=(18.*x.^17.*y)./(x.^2 + y.^2).^10 - (20.*x.^19.*y)
    ./(x.^2 + y.^2).^11;

% % %  Iterative form:
duydx=((VF_on_i-4).*x.^(VF_on_i-5).*y)./(x.^2 + y.^2).^((VF_on_i
    /2)-1) - ((VF_on_i-2).*x.^(VF_on_i-3).*y)./(x.^2 + y.^2).^(
    VF_on_i/2);


[e11,e22,e12]=DifferentStrainTensorsFun(duxdx,duydy,duxdy,duydx,
    NL_GEOM);

    end
end
```

CalcStress5.m

Function calculating IVW^* for any VF.

```
function [IVW_Ma_Li,IVW_Ma_Ca,IVW_Ma_Pla,IVW_Ma_Wall,IVW_Ma_Homo
    ] ...
      = CalcStress5(B11_vec,B22_vec,B12_vec ...
       ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec ...
       ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
       ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
       ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec ...
       ,e11,e22,e12,A_ma_vec)
[M,N]=size(e11);
for i=1:N
S11_Li=2.*B11_Li_vec.*e11(:,i);
S22_Li=2.*B22_Li_vec.*e22(:,i);
S12_Li=4.*B12_Li_vec.*e12(:,i);
IVW_Ma_Li(:,i)=A_ma_vec.*(S11_Li+S22_Li+S12_Li);

S11_Ca=2.*B11_Ca_vec.*e11(:,i);
S22_Ca=2.*B22_Ca_vec.*e22(:,i);
S12_Ca=4.*B12_Ca_vec.*e12(:,i);
IVW_Ma_Ca(:,i)=A_ma_vec.*(S11_Ca+S22_Ca+S12_Ca);

%  imagesc((BW_OuPla-BW_Lu-BW_Ca-BW_Li).*B11.*e11(:,i))
S11_Pla=2.*B11_Pla_vec.*e11(:,i);
S22_Pla=2.*B22_Pla_vec.*e22(:,i);
S12_Pla=4.*B12_Pla_vec.*e12(:,i);
IVW_Ma_Pla(:,i)=A_ma_vec.*(S11_Pla+S22_Pla+S12_Pla);

% imagesc(BW_Ou-BW_OuPla)
S11_Wall=2.*B11_Wall_vec.*e11(:,i);
S22_Wall=2.*B22_Wall_vec.*e22(:,i);
S12_Wall=4.*B12_Wall_vec.*e12(:,i);
IVW_Ma_Wall(:,i)=A_ma_vec.*(S11_Wall+S22_Wall+S12_Wall);

S11=2.*B11_vec.*e11(:,i);
S22=2.*B22_vec.*e22(:,i);
S12=4.*B12_vec.*e12(:,i);
IVW_Ma_Homo(:,i)=A_ma_vec.*(S11+S22+S12);
end
end
```

Rotations_B.m

Main file for calculating IVW^*, and rotating the coordinate system.

```
function [IVW_Pla,IVW_Li,IVW_Ca,IVW_Wall,IVW_Homo] ...
```

```
    =Rotations_B(B11_vec,B22_vec,B12_vec  ...
     ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec  ...
     ,B11_Li_vec,B22_Li_vec,B12_Li_vec  ...
     ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec  ...
     ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec...
     ,zi1_mid,zi2_mid,A_ma,VF_on,theta,PlotRotFigs,NL_GEOM)


A_ma_vec=A_ma(:);

x_vec=zi1_mid(:);
y_vec=zi2_mid(:);

% Rotation

cs = cosd(theta);
sn = sind(theta);

x_vec_new = x_vec .* cs - y_vec .* sn;
y_vec_new = x_vec .* sn + y_vec .* cs;

x_vec=x_vec_new;
y_vec=y_vec_new;

angle=theta;
% [B11_vec_R,B22_vec_R,B12_vec_R]=  ...
%       RotB_vec(B11_vec,B22_vec,B12_vec,angle);
[B11_vec,B22_vec,B12_vec]=  ...
     RotB_vec(B11_vec,B22_vec,B12_vec,angle);

[B11_Pla_vec,B22_Pla_vec,B12_Pla_vec]=  ...
     RotB_vec(B11_Pla_vec,B22_Pla_vec,B12_Pla_vec,angle);

[B11_Li_vec,B22_Li_vec,B12_Li_vec]=  ...
     RotB_vec(B11_Li_vec,B22_Li_vec,B12_Li_vec,angle);

[B11_Ca_vec,B22_Ca_vec,B12_Ca_vec]=  ...
     RotB_vec(B11_Ca_vec,B22_Ca_vec,B12_Ca_vec,angle);

[B11_Wall_vec,B22_Wall_vec,B12_Wall_vec]=  ...
     RotB_vec(B11_Wall_vec,B22_Wall_vec,B12_Wall_vec,angle);

```

```


if PlotRotFigs==1
figure;plot(x_vec,y_vec)
end


for i=1:length(VF_on)
VF_on_i=VF_on(i);
[e11(:,i),e22(:,i),e12(:,i)]=WorkingFieldsVF16_vec(x_vec,y_vec,
    VF_on_i,NL_GEOM);
end

[IVW_Ma_Li,IVW_Ma_Ca,IVW_Ma_Pla,IVW_Ma_Wall,IVW_Ma_Homo] ...
      = CalcStress5(B11_vec,B22_vec,B12_vec ...
       ,B11_Ca_vec,B22_Ca_vec,B12_Ca_vec ...
       ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
       ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
       ,B11_Wall_vec,B22_Wall_vec,B12_Wall_vec ...
       ,e11,e22,e12,A_ma_vec);

[M,N]=size(IVW_Ma_Pla);
for ii=1:N
IVW_Pla(ii,1)=nansum(IVW_Ma_Pla(:,ii));
IVW_Li(ii,1)=nansum(IVW_Ma_Li(:,ii));
IVW_Ca(ii,1)=nansum(IVW_Ma_Ca(:,ii));
IVW_Wall(ii,1)=nansum(IVW_Ma_Wall(:,ii));
IVW_Homo(ii,1)=nansum(IVW_Ma_Homo(:,ii));
end


end
```

RotB_vec.m

Function for rotation second order tensor $\underline{\underline{B}}$.

```
function [B11_vec_R,B22_vec_R,B12_vec_R]= ...
      RotB_vec(B11_vec,B22_vec,B12_vec,angle)
% angle=55;
B11=B11_vec;
B22=B22_vec;
B12=B12_vec;

```

```
sn=sind(angle);
cn=cosd(angle);

B11_vec_R=(cn.*((cn.*B11)-(sn.*B12)))-(sn.*((cn.*B12)-(sn.*B22))
    );
B12_vec_R=(sn.*((cn.*B11)-(sn.*B12)))+(cn.*((cn.*B12)-(sn.*B22))
    );
B22_vec_R=(sn.*((sn.*B11)+(cn.*B12)))+(cn.*((sn.*B12)+(cn.*B22))
    );

% Test if transposed B12=B21
% B12_2=(cn.*(sn.*B11+cn.*B12))-(sn.*(sn.*B12+cn.*B22));
% B12_vec_R-B12_2;
% nansum(ans(:))

end
```

RotNoLu.m

Function for rotation the lumen coordinates.

```
function [NoLuR]=RotNoLu(NoLu,theta)

cs = cosd(theta);
sn = sind(theta);

x_new = NoLu(2,:) .* cs - NoLu(3,:) .* sn;
y_new = NoLu(2,:) .* sn + NoLu(3,:) .* cs;

NoLuR=[NoLu(1,:);x_new;y_new];
%
% figure;plot(NoLu(2,:),NoLu(3,:),'.')
% figure;plot(NoLuR(2,:),NoLuR(3,:),'.')

end
```

F_Obj.m

```
function [F] =F_Obj(x0)
load('Vars_sum.mat')

F=(x0(1).*IVW_Pla+x0(2).*IVW_Li+x0(3).*IVW_Ca+x0(4).*IVW_Wall)-
    EVW;

```

```
F=F(:);
end
```

I.2. MATLAB code for real data

In this Appendix, the custom-built MATLAB code for the case study is given. Please note that functions that were already given in the previous section are not repeated. First, the main file is given, then all functions and files that are called upon in the main file are given.

VFM_RealData.m

The main file for using VFM on real data.

```
clear
clc
close all

% addpath NewData2
addpath NewData

addpath VFM_RESCRIPTED_VERSION3_FOR_MULTIPLE_MATLAB
addpath VFM_RESCRIPTED_VERSION3_FOR_MULTIPLE_MATLAB/functions

%%
MATLABPROCESSING
% MATLABPROCESSING2
DefineParameters_n4_Real

%%
FindLumen

%%
TimeVar=tic;
VFM_Real
Time=toc(TimeVar);


% save('RESULTS_NEW__DATA2__GRAD_BEFORE_MASK/80mmHg_VF_7_ROT_30
    ')

% DispVars
```

MATLABPROCESSING.m

File for processing the experimental data.

```
% clear all
% close all
% clc

% load('Data_pos148_pres340to5_001to034_finalRES.mat')

% importfile('MRI_S10_138_slice148_WITHcomponents.png')
%
% IM=MRI_S10_138_slice148_WITHcomponents;

% IM2=imread('MRI_S10_138_slice148_WITHcomponents.png');
IM=imread('S10_138_slice148.png');
% IM_G=rgb2gray(IM);
I64 = double(IM)/65535;
W=zeros(size(IM));
k=find(I64>0.58);
W(k)=I64(k);
% figure;imagesc(W)


[BW_C,Thressout]=edge(W,'canny');
Thressout(1)=Thressout(1);
Thressout(2)=Thressout(2);

[BW_C2]=edge(W,'canny',Thressout);
% figure;imagesc(BW_C2)

CH = bwconvhull(BW_C2,'objects');
% CH_2 = imfill(BW_C2,'holes');
% figure;imagesc(CH)

% CH2=imdilate(CH,true(0.1));
% figure;imagesc(CH2)
% CH(329,562)=0;
% CH(331,565)=0;
% CH(335,570)=0;
NIM=CH.*I64;
figure;imagesc(NIM)
%% Adventitia
% figure;imagesc(IM)
% figure;plot(sort(IM(:)))
% figure;imshow(IM)
```

```
A=zeros(size(IM));
% k=find(IM>33040);
% k=find(IM>33280);
% k=find(NIM>0.6285);
% k=find(NIM>0.6345);
% k=find(NIM>0.6476);
k=find(NIM>0.6476);
A(k)=NIM(k);
figure;imagesc(A)
[BW_C3]=edge(A,'canny',Thressout);
% CH2 = bwconvhull(BW_C3,'objects');
CH2 = imfill(BW_C3,'holes');

BW_Adventitia=CH-CH2;

%% Media
B=zeros(size(A));
k=find(A>0.661&A<0.7726);
B(k)=A(k);
% figure;imagesc(B)
Thressout2(1)=Thressout(1)+0.24;
Thressout2(2)=Thressout(2)+0.24;
[BW_C4]=edge(B,'canny',Thressout2);
% figure;imagesc(BW_C4)
% CH3 = bwconvhull(BW_C4,'objects');
CH3 = imfill(BW_C4,'holes');

BW_Media=(CH2-CH3);

%% Lipid
C=zeros(size(A));
% k=find(A>0.7955&A<0.84);
k=find(A>0.8142&A<0.84);
C(k)=A(k);
% figure;imagesc(C)
Thressout3(1)=Thressout(1)+0.32;
Thressout3(2)=Thressout(2)+0.32;
[BW_C5]=edge(C,'canny',Thressout3);
% figure;imagesc(BW_C5)

% CH4 = bwconvhull(BW_C5,'objects');
CH5=imfill(BW_C5,'holes');
```

```
BW_Lipid=CH5;

BW_Plaque=CH-BW_Adventitia-BW_Media-BW_Lipid;
%% Lumen
D=zeros(size(A));
E=(I64.*CH);
% k=find(A>0.8142&A<0.84);
% k=find(E>0.499&E<0.501);
k=find(E>0.480&E<0.520);
D(k)=E(k);
% figure;imagesc(D)

% Thressout4(1)=Thressout(1)+0.32;
% Thressout4(2)=Thressout(2)+0.32;
Thressout4(1)=Thressout(1)+0.8;
Thressout4(2)=Thressout(2)+0.8;
[BW_C6]=edge(D,'canny',Thressout4);
% figure;imagesc(BW_C6)
BW_Lumen=imfill(BW_C6,'holes');
% figure;imagesc(BW_Lumen)

BW_Plaque_LumenNotMasked=BW_Plaque;
BW_Plaque=BW_Plaque-BW_Lumen;
%%
% figure;imagesc(BW_Lipid)

% X(:,:)=xcoor(:,:,33);
% Y(:,:)=ycoor(:,:,33);

BW_All=(CH-BW_Lumen);

% figure;imagesc(BW_All)
BW_All = MaskAdjustment(BW_All);
% figure;imagesc(BW_All_New)
BW_Adventitia = MaskAdjustment(BW_Adventitia);
BW_Media = MaskAdjustment(BW_Media);
BW_Plaque = MaskAdjustment(BW_Plaque);
BW_Lipid = MaskAdjustment(BW_Lipid);
BW_Lumen = MaskAdjustment(BW_Lumen);

clearvars -except BW_All BW_Media BW_Adventitia BW_Plaque ...
    BW_Lumen BW_Lipid
```

```

load Data_pos148_pres10to160_001to024_finalRES;
% PressSteps=[1, 15, 19 21]; noRes=length(PressSteps);

xcoor = xcoor.*10^3;
ycoor = ycoor.*10^3;
lateralDisp=lateralDisp.*10^3;
axialDisp=axialDisp.*10^3;


%%% The displacements at 5 should be equal to 5-1 as below
% AA=axialDisp(:,:,5);
% figure;imagesc(AA);
% figure;imagesc(ycoor(:,:,5)-ycoor(:,:,1))
% AA=AA.*BW_Plaque;
% figure;imagesc(BW_Plaque)
% figure;imagesc(AA);

%%% Translating the coordinate origin to the center of the lumen

% Check left and right coordinate lumen in figure
% xxxx=xcoor(:,:,1);
% figure;imagesc(xxxx)
CoorR=8.305;
CoorL=6.022;
xTranslate=(CoorR+CoorL)/2;
% Check bottom and top coordinate lumen in figure
% yyyy=ycoor(:,:,1);
% figure;imagesc(yyyy)
CoorTop=5.173;
CoorBot=7.041;
yTranslate=(CoorBot+CoorTop)/2;
xcoor_trans=xcoor-xTranslate;
ycoor_trans=ycoor-yTranslate;

% Check if translation was done correctly
% xxxx_trans=xcoor_trans(:,:,1);
% yyyy_trans=ycoor_trans(:,:,1);
% xxxx_trans=xcoor_trans(:,:,24);
% yyyy_trans=ycoor_trans(:,:,24);
%
% figure;imagesc(xxxx_trans)
% figure;imagesc(yyyy_trans)

xcoor=xcoor_trans;
ycoor=ycoor_trans;

PressureVec=[10:5:100, 110:10:140,160];
```

MaskAdjustment.m

Function for adjusting the masks to the right size.

```
% Script to get the mask
% Original size of US in Nijmegen: 4320x511
% Original size of the mask: 814x1144 -> to crop because it also
    includes
% RF data and other stuff that is not necessary

function mask2 = MaskAdjustment(mask)

% FileName = 'US_"10_138_slice148.png';
% mask = imread(FileName);
% figure, imagesc(mask);

mask = mask(144:655, 235:790);
% figure, imagesc(mask);

%Conversion to logical values
% for i=1:length(mask(:,1))
%     for j=1:length(mask(1,:))
%         if mask(i,j) == 255
%             mask(i,j) = 1;
%         else
%             mask(i,j) = 0;
%         end
%     end
% end

%Resize to adapt to the US in Nijmegen and selection of the
    right area
mask = imresize(mask, [4320, 511]);
% figure, imagesc(mask);

mask = mask(241:4074, :);
%mask = mask(245:4295, :);
```

```
% figure , imagesc(mask);

mask = imresize(mask,[426, 511]);
%mask = imresize(mask,[451, 511]);
% figure;imagesc(mask)
mask = double(mask);
% figure; imagesc(mask);
% figure; imshow(mask)


% figure;imagesc(mask)
% figure;imshow(mask)
% mask(find(mask~=1))=0;
% figure;imagesc(mask)


mask3=round(mask);
% figure;imagesc(mask3)
for i=1:length(mask3(:,1))
    for j=1:length(mask3(1,:))
        if mask3(i,j) == 0
            mask3(i,j) = NaN;
        end
    end
end


mask2=mask3;
end
```

DefineParameters_n4_Real.m

Defining the input parameters for VFM.

```

% Choose pressure step
Pressure_mmHg=100;
Pressure=Pressure_mmHg/7500.6156130264;

% SwitchDilatedLu=0;%don't use dilated lumen
SwitchDilatedLu=1;%use dilated lumen

PixSwitch2=1;%Turn noisy pixels off
% PixSwitch2=0;%Don't turn noisy pixels off
% PixSwitch=1;%Turn noisy pixels off
PixSwitch=0;%Don't turn noisy pixels off

% MaskBefore=0;% use mask after calculating B11
MaskBefore=1;% use mask before calculating B11

Switch2Components=0;% Don't estimate 2 components
% Switch2Components=1;% Estimate 2 components

GAswitch=0;%Don't use GA
% GAswitch=1 % Use GA

translate_x=0;
translate_y=0;
% ROT_a=0;
% [x noise, y noise]
% noise=(0).*[1.8*10^(-3) 0.63*10^(-3)];
% noise=(16/5).*[1.8*10^(-3) 0.63*10^(-3)];
% noise=(16/1).*[1.8*10^(-3) 0.63*10^(-3)];
% noise=[0 0];
% VF_on=[1:11,16:20];
% VF_on=[1:11,16:17];
% VF_on=[1];
% VF_on=[2:5];
VF_on=[1:7];
% save('VF_on.mat','VF_on')
% j=0;
% j=1;
% j=5;
% j=6;
% j=12;
% j=100;
j=26;

StartRot=1;
EndNrOfRot= j;
MaxRot=180-(180/(j+1));

MaxRan=100; % number of random initial guesses


```

```
% MaxRot=360-(360/(j+1));

NL_GEOM=0; %Always set this to zero, as small strain definition
% has to be used for the virtual strain

%if NL_GEOM==1 % virtual Euler Amansi strain
% elseif NL_GEOM==0 % virtual Linear Strain
% elseif NL_GEOM==2 % virtual Green Lagrange strain



PlotRotFigs=0; % if ==1 plot geometries, if ==0 don't
% ClockIsChecked=0;
AngleEVW=90;
NrOfSteps=10;
StartInc=10;
EndInc=10;
```

FindLumen.m

File for defining the lumen.

```
% Find Lumen

% figure;imagesc(BW_All)
BW_All_bin=BW_All;
BW_All_bin(isnan(BW_All_bin)==1)=0;
% figure;imagesc(BW_All_bin)
[B,L,n,A] = bwboundaries(BW_All_bin,8);
figure;imagesc(BW_All_bin);hold on;
plot(B{1}(:,2),B{1}(:,1),'c');
plot(B{2}(:,2),B{2}(:,1),'r');
hold off
% Visual inspection, which one is the lumen?
% [B2,L,n,A] = bwboundaries(BW_Lumen,8);
% figure;imagesc(BW_Lumen);hold on;
% plot(B2{1}(:,2),B2{1}(:,1),'c');

LuCo=B{2};
AA=LuCo(:,2);
BB=LuCo(:,1);
LuCo=[AA BB]';

% Visual check
```

```
figure;plot(LuCo(1,:),LuCo(2,:),'r');hold on;
plot(B{2}(:,2),B{2}(:,1),'r*');

NoLuNR=[1:length(LuCo)];

NoLuUnDef_Pix_Cor=[NoLuNR;LuCo];
% CheckClockFig_Real(NoLuUnDef_Pix_Cor)

PressureStep=1;
% XX=BW_All.*xcoor(:,:,PressureStep);

XX=xcoor(:,:,PressureStep);
YY=ycoor(:,:,PressureStep);

NoLu_undef=zeros(size(NoLuUnDef_Pix_Cor));
NoLu_undef(1,:)=NoLuUnDef_Pix_Cor(1,:);
for i_lu=1:length(NoLuUnDef_Pix_Cor)
     NoLu_undef(2,i_lu)=XX(NoLuUnDef_Pix_Cor(3,i_lu),
        NoLuUnDef_Pix_Cor(2,i_lu));
     NoLu_undef(3,i_lu)=YY(NoLuUnDef_Pix_Cor(3,i_lu),
        NoLuUnDef_Pix_Cor(2,i_lu));


end

% close(figure(99))
% CheckClockFig_Real(NoLu_undef)

PressureStep=find(PressureVec==Pressure_mmHg);
XX_def=xcoor(:,:,PressureStep);
YY_def=ycoor(:,:,PressureStep);

NoLu=zeros(size(NoLuUnDef_Pix_Cor));
NoLu(1,:)=NoLuUnDef_Pix_Cor(1,:);
for i_lu2=1:length(NoLuUnDef_Pix_Cor)
     NoLu(2,i_lu2)=XX_def(NoLuUnDef_Pix_Cor(3,i_lu2),
        NoLuUnDef_Pix_Cor(2,i_lu2));
     NoLu(3,i_lu2)=YY_def(NoLuUnDef_Pix_Cor(3,i_lu2),
        NoLuUnDef_Pix_Cor(2,i_lu2));

end
close(figure(99))
```

```
CheckClockFig_Real(NoLu)
% figure;imagesc(BW_All)
% figure;plot(XX,YY,'.')
% figure;imagesc(XX);
%
% hold on;plot(NoLuUnDef_Pix_Cor(2,:),NoLuUnDef_Pix_Cor(3,:))

clearvars i_fig NoLuNR LuCo AA BB B n i_lu NoLuUnDef_Pix_Cor
```

VFM_Real.m

Main file containing the VFM-based approach for real data.

```
% VFM
% Calculate EVW
[EVW]=CalcEVW_Real(NoLu,Pressure,VF_on,AngleEVW);
incx=0.0275;
incy=0.0271;
BW_Lumen2=BW_Lumen;
BW_Lumen2(isnan(BW_Lumen2))=0;
BW=imdilate(BW_Lumen2,true(6));
% BW((BW==0))=NaN;
BW_All_New=BW_All-BW;
BW_All_New((BW_All_New==0))=NaN;
if PixSwitch2==1
      BW_All_New(140,286)=NaN;
      BW_All_New(141,288)=NaN;
      BW_All_New(141,287)=NaN;
      BW_All_New(141,286)=NaN;
      BW_All_New(142,292)=NaN;
      BW_All_New(142,291)=NaN;
      BW_All_New(171,220)=NaN;
      BW_All_New(142,286)=NaN;
end

zi1_mid=BW_All.*XX_def;
zi2_mid=BW_All.*YY_def;
if SwitchDilatedLu==1
zi1_mid=(BW_All_New).*XX_def;
zi2_mid=(BW_All_New).*YY_def;
end
% Calculate IVW
[B11_old,B12_old,B22_old]=CalcLCG(zi1_mid,zi2_mid,incx,incy);
```

```
% [B11_2,B12_2,B22_2]=CalcLCG((BW_Ou-BW_Lu).*zi1_mid,(BW_Ou-
    BW_Ca).*zi2_mid,incx,incy);
if PixSwitch==1
BW_pix=zeros(size(B11_old));
BW_pix(:,:)=1;
% BW_pix(BW_pix==0)=NaN;
% BW_pix(175,221)=0;
% BW_pix(178,220)=0;
% BW_pix(184,218)=0;

BW_pix(143,286)=NaN;


B11_old=B11_old.*BW_pix;
B22_old=B22_old.*BW_pix;
B12_old=B12_old.*BW_pix;
end


[B11_N,B12_N,B22_N]=CalcLCG(XX_def,YY_def,incx,incy);
B11=B11_N.*BW_All;
B22=B22_N.*BW_All;
B12=B12_N.*BW_All;

if MaskBefore==1
     B11=B11_old;
     B22=B22_old;
     B12=B12_old;
end

figure;imagesc(B11);title('B_{11}'); colorbar
figure;imagesc(B22);title('B_{22}'); colorbar
figure;imagesc(B12);title('B_{12}'); colorbar

%%

%%% Using vectors
% [A_ma]=Calc_A_ma(zi1_mid,zi1,zi2);
A_ma=zeros(size(zi1_mid));
A_ma(:)=incx*incy;

ii=0;
```

```


[IVW_Pla,IVW_Li,IVW_Ad,IVW_Media,IVW_Homo ...
      ,B11_vec,B22_vec,B12_vec ...
      ,B11_Ad_vec,B22_Ad_vec,B12_Ad_vec ...
      ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
      ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
      ,B11_Media_vec,B22_Media_vec,B12_Media_vec] ...
     =Rotations_A_Real(B11,B12,B22, ...
     BW_All,BW_Adventitia,BW_Lipid,BW_Media,BW_Plaque,zi1_mid,
         zi2_mid, ...
     A_ma,VF_on,PlotRotFigs,NL_GEOM);

if PlotRotFigs==1
hold on;plot(NoLu(2,:),NoLu(3,:),'r');hold off
end

ii=1;
for oo=StartRot:EndNrOfRot
theta=(oo/EndNrOfRot)*MaxRot;
Theta_vec(ii)=theta;

[IVW_Pla(:,ii+1),IVW_Li(:,ii+1),IVW_Ad(:,ii+1),IVW_Media(:,ii+1)
    ,IVW_Homo(:,ii+1)] ...
     =Rotations_B_Real(B11_vec,B22_vec,B12_vec ...
      ,B11_Ad_vec,B22_Ad_vec,B12_Ad_vec ...
      ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
      ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
      ,B11_Media_vec,B22_Media_vec,B12_Media_vec...
      ,zi1_mid,zi2_mid,A_ma,VF_on,theta,PlotRotFigs,NL_GEOM);


[NoLuR]=RotNoLu(NoLu,theta);

if PlotRotFigs==1
hold on;plot(NoLuR(2,:),NoLuR(3,:),'r')
hold off
end

[EVW(:,(ii+1))]=CalcEVW_Real(NoLu,Pressure,VF_on,AngleEVW);


```

```

ii=ii+1;
end

%% Minimization

% EVW(2)=-EVW(2);
save('temp/Vars_sum.mat','IVW_Pla','IVW_Li','IVW_Ad','IVW_Media'
    ,'EVW' ...
     ,'VF_on')

% Initial Guess
% x_ini=x0;
% x_ini=[1 1 1 1];
% x_ini=[plaque,lipid,adventitia,media]
x_ini=[0.2 0.00001 0.25];

[F] =F_Obj_Real_3cp(x_ini);
% [F] =F_Obj_Real2(x_ini);

% [F_ma]=F_Obj_mat(x0);
% [F3] =F_Obj3(x0)
SumF=sum(abs(F));

%
% Algorithm

%% Trust region reflective
options.Algorithm='trust-region-reflective';
% options.OptimalityTolerance=1.0000e-020;

% options = optimoptions(@lsqnonlin,'Algorithm','trust-region-
    reflective',...
%      'OptimalityTolerance',1.0000e-1060,...
%      'StepTolerance', 1.0000e-1000, ...
% 'MaxFunctionEvaluations',10000, ...
% 'MaxIterations',10000);

[c_esti_TR_3cp] = lsqnonlin(@F_Obj_Real_3cp,x_ini,[0 0 0],[],
    options);

if Switch2Components==1
```

```
x_ini=[0.2  0.25];
[c_esti_TR_2cp] = lsqnonlin(@F_Obj_Real2,x_ini,[0  0],[],options)
    ;
end
disp('Calculated values using trust region reflective')

%% Choose random initial guesses
TimeVar3=tic;
options3.Algorithm='trust-region-reflective';
options3.Display='off';
% MaxRan=100;
% c_esti_TR_Rv=zeros(MaxRan,4);
c_esti_TR_Rv=zeros(MaxRan,2);
for  rrr=1:MaxRan
x_ini2=rand(1,3);
[c_esti_TR_Rv_3cp(rrr,:)] = lsqnonlin(@F_Obj_Real_3cp,x_ini2,[0
    0  0],[],options3);

if  Switch2Components==1
x_ini2=rand(1,2);
[c_esti_TR_Rv_2cp(rrr,:)] = lsqnonlin(@F_Obj_Real2,x_ini2,[0
    0],[],options3);
end

end
% If only trying to estimate wall and intima so Real2
% c_esti_TR_Rv(:,2)=0;
% c_esti_TR_Rv(:,4)=0;

% ER=(c_esti_TR_Rv-x0)./x0;
% ER=abs(ER);
% [MaxER]=max(ER);
% ER_Pla_m=MaxER(1);
% ER_Li_m=MaxER(2);
% ER_Ca_m=MaxER(3);
% ER_Wall_m=MaxER(4);
% K=find(ER(:,1)==MaxER(1));
% L=find(ER(:,2)==MaxER(2));
% M=find(ER(:,3)==MaxER(3));
% N=find(ER(:,4)==MaxER(4));
%
% c_esti_TR_R(1)=c_esti_TR_Rv(K(1),1);
% c_esti_TR_R(2)= c_esti_TR_Rv(L(1),2);
% c_esti_TR_R(3)= c_esti_TR_Rv(M(1),3);
% c_esti_TR_R(4)= c_esti_TR_Rv(N(1),4);
figure;plot(c_esti_TR_Rv_3cp);title('100 random initial guesses'
    )
legend('Plaque','Lipid','Wall')
c_esti_TR_Random_Mean=mean(c_esti_TR_Rv_3cp);
TimeRandomIni=toc(TimeVar3);
disp('Calculated values using trust region reflective')
disp('With 100 random initial guesses')
%% Genetic algorithm
if GAswitch==1
TimeVar2=tic;
disp('Calculating values using genetic algorithm....')
% options2.PopulationSize=200;
% options2=optimoptions(@ga,'PopulationSize',200,'MaxGenerations
    ', Inf,'FunctionTolerance',1E-100);
c_esti_ga_4cp = ga(@F_Obj2_Real,4,[],[],[],[],[0 0 0 0],[],[]);
if Switch2Components==1
c_esti_ga_2cp = ga(@F_Obj2_Real2,2,[],[],[],[],[0 0],[],[]);
end
TimeGa=toc(TimeVar2);
disp('Calculated values using genetic algorithm')
end

%% Multiobjective Genetic Algorithm
% TimeVar3=tic;
% optionsmoga.DistanceMeasureFcn = {@distancecrowding,'genotype
    '};
% optionsmoga.ParetoFraction = 0.5;
% FitnessFunction = @F_Obj; % Function handle to the fitness
    function
% numberOfVariables = 4; % Number of decision variables
% lb = [0 0 0 0]; % Lower bound
% ub = []; % Upper bound
% A = []; % No linear inequality constraints
% b = []; % No linear inequality constraints
% Aeq = []; % No linear equality constraints
% beq = []; % No linear equality constraints
% [c_esti_moga,Fval,exitFlag,Output] = gamultiobj(
    FitnessFunction,numberOfVariables,A, ...
%     b,Aeq,beq,lb,ub,optionsmoga);
% TimeMoGa=toc(TimeVar3);

%%

% Homogeneous approach
c_homogeneous=EVW./IVW_Homo;


% Some figures for analyses
if PlotRotFigs==1
for tt=1:(1*((EndNrOfRot+1)-(StartRot-1)))
    figure(tt)
    hold on
    axis([-6 6 -6 6])
    hold off
end
end
%%% Looking at the objective function evaluated at a
%%% Specific value
% figure;plot(F)
%
% % FF=[F(1:length(VF_on));
% % F(length(VF_on)+1:2*length(VF_on));
% % F(2*length(VF_on)+1:3*length(VF_on))];
% %
%
% %%% analyse per angle
% if EndNrOfRot>0
% % GG=1;
% % FF(GG,:)=F(1:length(VF_on));
% % for GG=1:EndNrOfRot
% % FF(GG+1,:)=F(((GG*length(VF_on))+1):(GG+1)*length(VF_on));
% % end
% Theta_vec=[0 Theta_vec];
% figure;plot(Theta_vec,F_ma)
% hold on
% for qq=1:length(VF_on)
%     strng1=num2str(qq);
%     strng2='Virtual Field ';
%     strng3{qq,:}=[strng2 strng1];
%     text(Theta_vec(end), F_ma(qq,end),strng1)
% %     hold on
% % plot(Theta_vec(:),FF(:,qq))
% end
% legend(strng3)
% % set(gca, 'XTick', Theta_vec)
% hold off
% end
% figure;
% tic
% % Genetic algorithm
% x2 = ga(@F_Obj3,4,[],[],[],[],[0 0 0 0],[])
% toc
```

Rotations_A_Real.m

Main file for calculating IVW^* for the unrotated coordinate system.

```
function [IVW_Pla,IVW_Li,IVW_Ad,IVW_Media,IVW_Homo ...
      ,B11_vec,B22_vec,B12_vec ...
      ,B11_Ad_vec,B22_Ad_vec,B12_Ad_vec ...
      ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
      ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
      ,B11_Media_vec,B22_Media_vec,B12_Media_vec] ...
     =Rotations_A_Real(B11,B12,B22, ...
     BW_All,BW_Adventitia,BW_Lipid,BW_Media,BW_Plaque,zi1_mid,
         zi2_mid, ...
     A_ma,VF_on,PlotRotFigs,NL_GEOM)

B11_Pla=BW_Plaque.*B11;
B11_Li=BW_Lipid.*B11;
B11_Ad=BW_Adventitia.*B11;
B11_Media=BW_Media.*B11;

B22_Pla=BW_Plaque.*B22;
B22_Li=BW_Lipid.*B22;
B22_Ad=BW_Adventitia.*B22;
B22_Media=BW_Media.*B22;

B12_Pla=BW_Plaque.*B12;
B12_Li=BW_Lipid.*B12;
B12_Ad=BW_Adventitia.*B12;
B12_Media=BW_Media.*B12;



```

```

% create vectors,
B11_Pla_vec=B11_Pla(:);
B11_Li_vec=B11_Li(:);
B11_Ad_vec=B11_Ad(:);
B11_Media_vec=B11_Media(:);
B22_Pla_vec=B22_Pla(:);
B22_Li_vec=B22_Li(:);
B22_Ad_vec=B22_Ad(:);
B22_Media_vec=B22_Media(:);
B12_Pla_vec=B12_Pla(:);
B12_Li_vec=B12_Li(:);
B12_Ad_vec=B12_Ad(:);
B12_Media_vec=B12_Media(:);

B11_vec=B11(:);
B22_vec=B22(:);
B12_vec=B12(:);

A_ma_vec=A_ma(:);

x_vec=zi1_mid(:);
y_vec=zi2_mid(:);

if PlotRotFigs ==1
figure;plot(x_vec,y_vec)
end


for i=1:length(VF_on)
VF_on_i=VF_on(i);
[e11(:,i),e22(:,i),e12(:,i)]=WorkingFieldsVF16_vec(x_vec,y_vec,
    VF_on_i,NL_GEOM);
end

[IVW_Ma_Li,IVW_Ma_Ad,IVW_Ma_Pla,IVW_Ma_Media,IVW_Ma_Homo]  ...
      = CalcStress5(B11_vec,B22_vec,B12_vec  ...
       ,B11_Ad_vec,B22_Ad_vec,B12_Ad_vec  ...
       ,B11_Li_vec,B22_Li_vec,B12_Li_vec  ...
       ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec  ...
       ,B11_Media_vec,B22_Media_vec,B12_Media_vec  ...
       ,e11,e22,e12,A_ma_vec);
```

```

[M,N]=size(IVW_Ma_Pla);
for ii=1:N
IVW_Pla(ii,1)=nansum(IVW_Ma_Pla(:,ii));
IVW_Li(ii,1)=nansum(IVW_Ma_Li(:,ii));
IVW_Ad(ii,1)=nansum(IVW_Ma_Ad(:,ii));
IVW_Media(ii,1)=nansum(IVW_Ma_Media(:,ii));
IVW_Homo(ii,1)=nansum(IVW_Ma_Homo(:,ii));
end


end
```

Rotations_B_Real.m

Main file for calculating IVW^{*}, and rotating the coordinate system.

```
function [IVW_Pla,IVW_Li,IVW_Ad,IVW_Media,IVW_Homo] ...
      =Rotations_B_Real(B11_vec,B22_vec,B12_vec ...
       ,B11_Ad_vec,B22_Ad_vec,B12_Ad_vec ...
       ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
       ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
       ,B11_Media_vec,B22_Media_vec,B12_Media_vec...
       ,zi1_mid,zi2_mid,A_ma,VF_on,theta,PlotRotFigs,NL_GEOM)


A_ma_vec=A_ma(:);

x_vec=zi1_mid(:);
y_vec=zi2_mid(:);

cs = cosd(theta);
sn = sind(theta);

x_vec_new = x_vec .* cs - y_vec .* sn;
y_vec_new = x_vec .* sn + y_vec .* cs;

x_vec=x_vec_new;
y_vec=y_vec_new;

angle=theta;
% [B11_vec_R,B22_vec_R,B12_vec_R]= ...
%        RotB_vec(B11_vec,B22_vec,B12_vec,angle);
[B11_vec,B22_vec,B12_vec]= ...
```

```
    RotB_vec(B11_vec,B22_vec,B12_vec,angle);

[B11_Pla_vec,B22_Pla_vec,B12_Pla_vec]= ...
    RotB_vec(B11_Pla_vec,B22_Pla_vec,B12_Pla_vec,angle);

[B11_Li_vec,B22_Li_vec,B12_Li_vec]= ...
    RotB_vec(B11_Li_vec,B22_Li_vec,B12_Li_vec,angle);

[B11_Ad_vec,B22_Ad_vec,B12_Ad_vec]= ...
    RotB_vec(B11_Ad_vec,B22_Ad_vec,B12_Ad_vec,angle);

[B11_Media_vec,B22_Media_vec,B12_Media_vec]= ...
    RotB_vec(B11_Media_vec,B22_Media_vec,B12_Media_vec,angle);



if PlotRotFigs==1
figure;plot(x_vec,y_vec)
end


for i=1:length(VF_on)
VF_on_i=VF_on(i);
[e11(:,i),e22(:,i),e12(:,i)]=WorkingFieldsVF16_vec(x_vec,y_vec,
   VF_on_i,NL_GEOM);
end

[IVW_Ma_Li,IVW_Ma_Ca,IVW_Ma_Pla,IVW_Ma_Wall,IVW_Ma_Homo] ...
    = CalcStress5(B11_vec,B22_vec,B12_vec ...
     ,B11_Ad_vec,B22_Ad_vec,B12_Ad_vec ...
     ,B11_Li_vec,B22_Li_vec,B12_Li_vec ...
     ,B11_Pla_vec,B22_Pla_vec,B12_Pla_vec ...
     ,B11_Media_vec,B22_Media_vec,B12_Media_vec ...
     ,e11,e22,e12,A_ma_vec);

[M,N]=size(IVW_Ma_Pla);
for ii=1:N
IVW_Pla(ii,1)=nansum(IVW_Ma_Pla(:,ii));
IVW_Li(ii,1)=nansum(IVW_Ma_Li(:,ii));
IVW_Ad(ii,1)=nansum(IVW_Ma_Ca(:,ii));
IVW_Media(ii,1)=nansum(IVW_Ma_Wall(:,ii));
IVW_Homo(ii,1)=nansum(IVW_Ma_Homo(:,ii));
```

```
end

end
```

CalcEVW_Real.m

Function for calculating EVW.

```
function  [EVW]=CalcEVW_Real(NoLu,Pressure,VF_on,AngleEVW)

external_VP=zeros(length(VF_on),1);
external_VP_i=zeros(length(VF_on),(length(NoLu)-1));

% CheckClockFig
angle=AngleEVW;

for  uuu=1:length(VF_on)
     VF_on_i=VF_on(uuu);


% Calculating
for  i = 1:(length(NoLu)-1)
vector = NoLu(2:3,i+1)- NoLu(2:3,i);
[px, py]=rotating(vector(1),vector(2),angle);
normal=[px py]';
lnorm=sqrt((normal(1)^2) + (normal(2)^2));
lvector=sqrt((vector(1)^2)+(vector(2)^2));
% lvector_vector(i)=lvector;
normal=normal/lnorm;
% normalvector(i,:)=normal;
PositionMiddle=NoLu(2:3,i)+ (vector/2);
t=Pressure.*normal;
% t_vector(i,:)=t;
x=PositionMiddle(1);
y=PositionMiddle(2);
Theta=acos(x/sqrt((x^2)+(y^2)));
if  VF_on_i==1
     external_VP_i(uuu,i) = lvector*(((x*t(1))+(y*t(2)))/((x^2)+(
        y^2)));
end
if  VF_on_i==2
     external_VP_i(uuu,i) = lvector*((((x^3)*t(1))+(((x^2)*y)*t
        (2)))/(((x^2)+(y^2))^2));
end
```

```
if VF_on_i==3
   external_VP_i(uuu,i) = lvector*((((x*(y^2))*t(1))+((y^3)*t(2)
      ))/(((x^2)+(y^2))^2));
end
if VF_on_i==4
    external_VP_i(uuu,i) = lvector*((((x^5)*t(1))+(((x^4)*y)*t
       (2)))/(((x^2)+(y^2))^3));
end
if VF_on_i==5
    external_VP_i(uuu,i) = lvector*((((x*(y^4))*t(1))+((y^5)*t
       (2)))/(((x^2)+(y^2))^3));
end
if VF_on_i==6
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^6);
end
if VF_on_i==7
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^6);
end
if VF_on_i==8
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^8);
end
if VF_on_i==9
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^8);
end
if VF_on_i==10
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^10);
end
if VF_on_i==11
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^10);
end
if VF_on_i==12
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^2).*(
       cos(Theta).^2);
end
if VF_on_i==13
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^4).*(
       cos(Theta).^4);
end
if VF_on_i==14
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^6).*(
       cos(Theta).^6);
end
```

```
if VF_on_i==15
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^8).*(
       cos(Theta).^8);
end
if VF_on_i==16
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^12);
end
if VF_on_i==17
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^12);
end
if VF_on_i>17
    iseven = rem(VF_on_i, 2) == 0;
    if iseven ==0
    external_VP_i(uuu,i) = external_VP_i(i).*(sin(Theta).^(
       VF_on_i-5));
    elseif iseven ==1
    external_VP_i(uuu,i) = external_VP_i(i).*(cos(Theta).^(
       VF_on_i-4));

    end
end

external_VP(uuu,1) = external_VP(uuu,1) + external_VP_i(uuu,i);


end




% EVW=external_VP;
end
for zz=1:length(VF_on)
   EVW(zz,1)= nansum(external_VP_i(zz,:));
end
if EVW<0
     EVW=-EVW;
end



end
```

F_Obj_Real_3cp.m

File containing the objective function.

```
function [F] =F_Obj_Real_3cp(x0)
addpath temp
load('Vars_sum.mat')

F=(x0(1).*IVW_Pla+x0(2).*IVW_Li+x0(3).*(IVW_Ad+IVW_Media))-EVW;

F=F(:);
end
```

F_Obj_Real2.m

File containing the objective function.

```
function [F] =F_Obj_Real2(x0)
addpath temp
load('Vars_sum.mat')

X0(1)=x0(1);
X0(3)=x0(2);
% X0(2)=0.000001;
X0(2)=0.83/1000;
% X0(4)=0;
F=(X0(1).*IVW_Pla+X0(2).*IVW_Li+X0(3).*(IVW_Ad+IVW_Media))-EVW;

F=F(:);
end
```