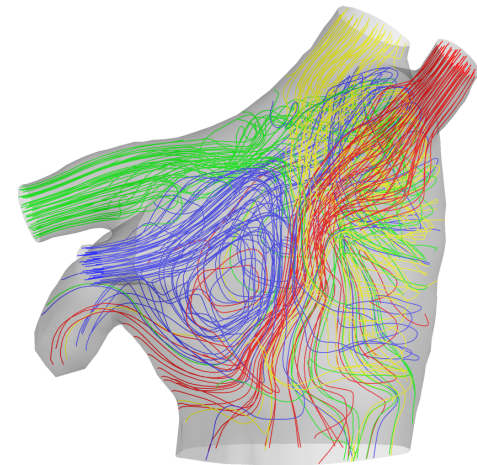


Vegard Slettaahjell Skjefstad

Influence of the Dynamic Motion of the Atrial Wall on the Hemodynamics in the Human Left Atrium: A Computational Fluid Dynamics Study

July 2020





Norwegian University of
Science and Technology

Influence of the Dynamic Motion of the Atrial Wall on the Hemodynamics in the Human Left Atrium: A Computational Fluid Dynamics Study

Vegard Slettahjell Skjefstad

Master of Mechanical Engineering

Submission date: July 2020

Supervisor: Victorien Emile Prot [IV]

Co-supervisor: Sigrid Kaarstad Dahl [SINTEF]

Norwegian University of Science and Technology
Department of Structural Engineering

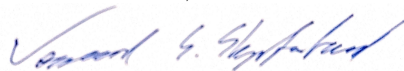
Acknowledgments

This Master Thesis was written at the Department of Structural Engineering at the Norwegian University of Science and Technology (NTNU) in collaboration with the Department of Health Research at SINTEF Digital. It was concluded during the spring semester of 2020 and was supervised by Associate Professor Victorien Emile Prot and co-supervised by Research Scientist Sigrid Kaarstad Dahl.

First of all, I would like to thank my supervisor Victorien Emile Prot for his contribution to the project with his expertise within biomechanics, and always having an interest in the project's progress and my well being during these troubling months. I would also like to express immense gratitude to my co-supervisor Sigrid Kaarstad Dahl for contributing with her vast knowledge within computational fluid dynamics within the field of biomechanics, and without whom this project would not exist. I would also like to express gratitude to PhD candidate Rasmus Hvid at Technical University of Denmark (DTU) for allowing me to use his MR recordings and valuable discussion of how to process these. Gratitude also goes to Marius Eriksen at St. Olavs Hospital, Trondheim for performing high-quality MR recordings.

Last but not least, I would like to thank my friends and loved ones for their support and motivation during my study, and especially during these last months.

Trondheim, 30.06.2020



Vegard Slettahjell Skjefstad

Abstract

Cardiovascular Diseases (CVDs) are the leading cause of death in Europe, where Atrial Fibrillation (AF) is one of the major causes, and the prevalence is expected to increase for the coming years. Demand for better diagnostic and treatment of AF is therefore increasing. Computational Fluid Dynamics (CFD) is viewed as an essential contributor to understanding the hemodynamics in the cardiovascular system and how different CVDs impact the blood flow. CFD analysis of the Left Atrium (LA), the most critical of the atria, have not yet reached clinical practice due to several model uncertainties. One of these uncertainties is the necessity of having a dynamic model of the LA or if a static model is sufficient, which this thesis aims to answer.

From Magnetic Resonance Imaging (MRI), a patient-specific LA geometry has been segmented for 25 time-steps in the cardiac cycle. The dynamic mesh in the CFD study is created by using image registration between the segmented volumes, and a displacement field is extracted and applied to the surface mesh of the LA. Results from the simulations imply very different flow characteristics in the intra-atrial flow between the static and dynamic cases. Compared to measured intra-atrial flow field in other studies, the dynamic case shows more agreement. The static case also shows higher tendencies of blood stasis where the risk of thrombus formation may occur. Two different geometrical static cases were also compared and showed little difference in their characteristics. The conclusion is therefore that a dynamic LA model is recommended for LA CVD studies. If a static model is used, there is no substantial difference in which time-step the geometry is extracted from.

The results do, however, exhibit difficulties in their convergence and require further study in order to create a model accurate enough to be used in clinical practice. The dynamic model presented also have areas of improvement concerning mesh quality.

Sammendrag

Hjerte- og karsykdommer er den ledende dødsårsaken i Europa, der atrieflimmer (AF) er en av de viktigste årsakene, og utbredelsen forventes å øke de kommende årene. Eterspørselen etter bedre diagnostikk og behandling av AF vil derfor øke. Numerisk fluiddynamikk (CFD) blir sett på som en viktig bidragsyter til å forstå hemodynamikken i det kardiovaskulære systemet og hvordan forskjellige hjerte- og karsykdommer påvirker blodstrømmen. CFD-analyse av venstre atrium (LA), den mest kritiske av atriene, har enda ikke nådd klinisk praksis på grunn av flere usikkerheter i modellen. En av disse usikkerhetene er nødvendigheten av å ha en dynamisk modell av LA eller hvis en statisk modell er tilstrekkelig, som denne oppgaven tar i sikte for å svare på.

Fra magnetisk resonansavbildning (MRI) har en pasientspesifikk LA-geometri blitt segmentert i 25 tidstrinn av hjertesyklusen. Det dynamiske meshet i CFD-studien er opprettet ved å bruke bilderegistrering mellom de segmenterte volumene, og et forskyvningsfelt blir generert og påført overflatemeshet. Resultater fra simuleringene innebærer veldig forskjellige strømmingsegenskaper i den intraatriale strømmingen mellom de statiske og dynamiske tilfellene. Det dynamiske tilfellet viser mer enighet med målt intraatriale strømningsfelt i andre studier. Det statiske tilfellet viser også høyere tendenser til blodstase der risikoen for trombedannelse kan oppstå. To forskjellige geometriske statiske tilfeller ble også sammenlignet og viste liten forskjell i deres egenskaper. Konklusjonen er derfor at en dynamisk LA-modell anbefales for LA CVD-studier. Hvis en statisk modell brukes, er det ingen vesentlig forskjell i hvilket tidstrinn geometrien segmenteres fra.

Resultatene viser imidlertid vanskeligheter med deres konvergens og krever videre studier for å lage en modell nøyaktig nok til å bli brukt i klinisk praksis. Den presenterte dynamiske modellen har også forbedringsområder når det gjelder meshkvalitet.

Contents

Acknowledgments	iii
Abstract	v
Sammendrag	vii
Contents	ix
Acronyms	xiii
1 Introduction	1
1.1 Scope and Aim	2
1.2 Approach	2
1.3 Structure	2
2 Heart Anatomy and Physiology	5
2.1 Left Atrium and Pulmonary Veins	6
2.2 Left Ventricle and Aorta	7
2.3 Mitral Valve	8
2.4 Cardiac cycle	8
2.5 Hemodynamics	11
2.6 Cardiovascular diseases	12
3 Litterature Review	15
3.1 Cardiac Imaging Techniques	15
3.2 Left Ventricle	16
3.3 Left Atrium	16
3.4 Pulmonary Veins	17
3.5 Mitral Valve	18
3.6 Hemodynamics	19
3.7 Discussion	19

4	Simulation Model Methodology	23
4.1	MRI Processing	23
4.2	Segmentation	25
4.3	Static Mesh	30
4.4	Dynamic Mesh	31
4.5	Fluent Dynamic Mesh	36
4.6	Boundary Conditions and Physics Continuum	39
4.6.1	Wall Deformation	39
4.6.2	Inlet Condition Dynamic Case	40
4.6.3	Inlet Condition Static Case	45
4.7	Fluent Solver	47
4.7.1	Discretisation Theory	47
4.7.2	Fluent Solver Settings	50
4.8	Inlet and Outlet Extension	51
4.8.1	Inlet Extension	51
4.8.2	Outlet Extension	56
5	Simulation results and discussion	59
5.1	Model Verification	60
5.2	Streamlines	65
5.3	Vortex Structures	69
5.4	Wall Shear Stress	72
5.5	Results Discussion	74
5.6	Simulation Model Discussion	76
5.6.1	Boundary Conditions	76
5.6.2	Segmentation	76
5.6.3	MV Modelling	77
5.6.4	Dynamic Mesh	77
6	Conclusion and Further Work	79
	Bibliography	81
A	Mesh	85
A.1	Mesh settings and statistics	85

A.2	Dynamic mesh settings	86
B	Fluent UDFs	87
B.1	Dynamic mesh	87
B.2	Polynomial coefficients	90
B.3	Mass flow inlet	91
C	MATLAB Scripts	95
C.1	Adjust nodes	95
C.2	Cubic spline interpolation	97
C.3	PV flow adjustment and spline interpolation	100

Acronyms

2CH Two-Chamber.

4CH Four-Chamber.

AF Atrial Fibrillation.

CFD Computational Fluid Dynamics.

CT Computed Tomography.

CVDs Cardiovascular Diseases.

DV Doppler Velocimetry.

ECG Electrocardiogram.

FSI Fluid-Structure Interaction.

LA Left Atrium.

LAA Left Atrial Appendage.

LES Large-Eddy Simulation.

LIPV Left Inferior Pulmonary Vein.

LSPV Left Superior Pulmonary Vein.

LV Left Ventricle.

MA Mitral Annulus.

MRI Magnetic Resonance Imaging.

MV Mitral Valve.

PVs Pulmonary Veins.

RA Right Atrium.

RBC Red Blood Cells.

Re Reynolds Number.

RIPV Right Inferior Pulmonary Vein.

RMSE Root Mean Square Error.

RSPV Right Superior Pulmonary Vein.

RV Right Ventricle.

SAX Short Axis.

SR Sinus Rhythm.

STL Stereolithography.

UDF User Defined Function.

US Ultrasound.

WSS Wall Shear Stress.

Chapter 1

Introduction

In 2017, CVDs was the leading cause of death in Europe, where it counted for 45% of all deaths [1]. AF is one of the major causes of CVDs, where AF causes structural changes in the atrial myocardium and blood stasis increases the risk of thrombus formation. The number of patients with AF is expected to rise steeply for the coming years [2]. It is therefore crucial to investigate better techniques for diagnostic and treatment of AF.

Computational methods have had a substantial increase in interest over the last two decades and, during the last decade, highly accurate models have been created. The significant benefit of computational methods is that it is a non-invasive technique able to give fundamental insight into the behaviour of blood and tissue. Computational cardiovascular methods are beginning to enter clinical practice, most notably *Heartflow*® *FFRCT* which simulates blood flow in the coronary arteries [3].

Computational methods are also used to evaluate the effects of e.g. AF by utilising CFD. The CFD models are able to determine how different atrial geometry, movement, inflow and outflow impacts the intra-atrial flow and evaluate the risk of CVDs to occur. These methods have not yet reached clinical practice as there are uncertainties regarding how accurate the models are. One of these uncertainties is the necessity of a dynamic model in order to account for the cardiac movement of the atrium. A comparison of a static and dynamic left atrial CFD model has, to the author's knowledge, not yet been performed.

1.1 Scope and Aim

The scope of this thesis is the CFD modelling of the LA of the heart as the left part is the most common area of heart failure [4]. A patient-specific dynamic model taking into account the movement of the atrial wall will be made from MRI and evaluated against a static model. The aim is to determine how the intra-atrial flow vary in the dynamic and static models and determine which of these models should be used for further investigation of LA CVDs and their causes, such as AF.

1.2 Approach

By studying anatomy and physiology and other CFD studies of the heart, details necessary for the CFD model will be determined, and which to be neglected. By further improving upon unpublished work by the author [5], a static and dynamic mesh is made from MRI segmentation of the LA. In order to determine how boundary conditions impact the model, test cases are performed to improve the CFD model. Currently, there are no available measurements to validate the results, so instead characteristic differences of the intra-atrial flow between the dynamic and static cases will be used for the conclusion of this thesis.

1.3 Structure

The thesis consists of six chapters, the first being the introduction, and an Appendix. Relevant theory will be presented where it is suitable.

Second chapter introduces a brief overview of relevant heart anatomy and physiology.

Third chapter investigate recent studies within computational cardiac modelling, from image techniques to CFD, and discuss the results to determine relevant details for the CFD model.

Fourth chapter presents the methodology of how the CFD model for a static and dynamic left atrium is made, from imaging techniques to a complete simulation model.

Fifth chapter presents the results of the simulations, along with a discussion of the results and the CFD model.

Sixth chapter presents the conclusion of this thesis and further work to be performed in order to improve the CFD model and results.

Appendix consists of simulation model settings and scripts central to the model.

Chapter 2

Heart Anatomy and Physiology

The heart is the driving force of the cardiovascular system and can be split up into four chambers; the right and left *atria* and the right and left *ventricles*. Blood flows from the right ventricle into the *pulmonary trunk*, through the lungs into the *pulmonary veins* and into the left atrium. This is called the *pulmonary circulation*, where the blood receives oxygen in the lungs. From the left atrium, the blood flows into the left ventricle and then into the *aorta*, which distributes the blood throughout the entire body before returning to the right atrium. This is called the *systemic circulation*, where blood transports oxygen to the body.

Between the atria and ventricles are valves that allow blood to pass unidirectionally from the atria to the ventricles. These valves are called the *tricuspid* and *mitral valve* for the right and left part of the heart, respectively, and also serves as the connection between the pulmonary and systemic circulation. Blood flowing from the right ventricle into the pulmonary trunk and from the left ventricle to the aorta are controlled by the pulmonary and aortic valves, respectively. Figure 2.1 shows an overview of the heart anatomy and the cardiovascular system.

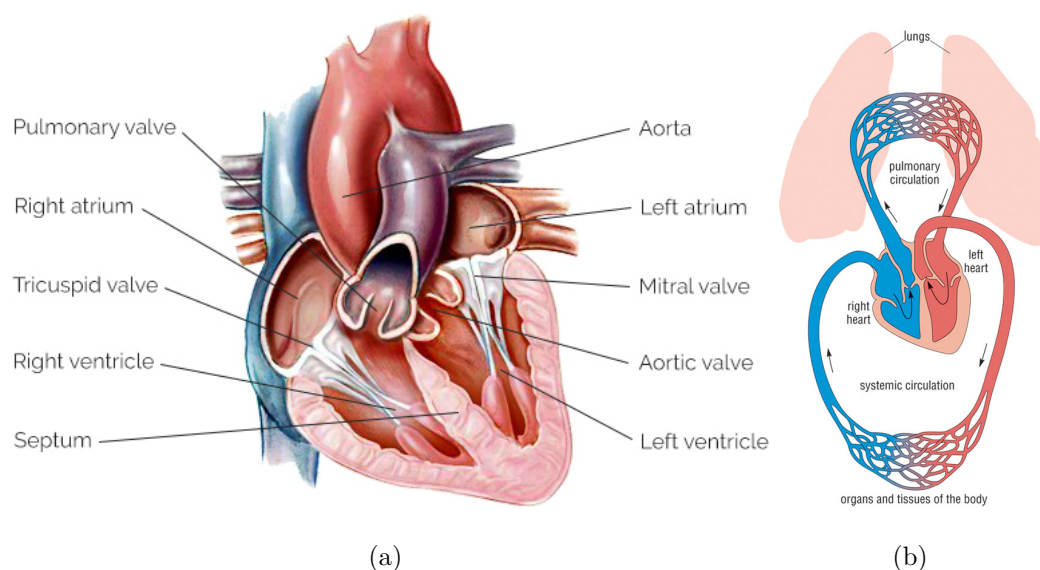


Figure 2.1: Heart anatomy and cardiovascular system. (a) shows an overview of the different heart chambers and valves. (b) shows an overview of the pulmonary and systemic circulation. (a) is from [6] and (b) from [7].

The following subsections will go into detail of the anatomy of the left part of the heart and its cardiac cycle as the right part is not within the scope of this thesis. The left ventricle is neither within the scope but has a significant impact on how the left atrium behaves. The anatomy description is based on the books Textbook of the Human Anatomy (1976) by W.J. Hamilton [8], Atlas of Heart Anatomy and Development (2014) by F.M. Filipoiu [9], Guyton and Hall Textbook of Medical Physiology (2019) by J.E. Hall [10] and Cardiovascular Physiology Concepts (2012) by R.E. Klabunde [11].

2.1 Left Atrium and Pulmonary Veins

The interior of the LA is for the most part smooth except for the Left Atrial Appendage (LAA), also called *auricle*, a small finger-like protrusion on the lower side of the LA with a considerable variety of size and shape [12]. The LA and LAA contain muscle fibres which allow it to contract and expand, influencing the pressure and volume of the blood within. The Pulmonary Veins (PVs) are connected to the upper part of the LA. It is most common to have four PVs, two at the anterior, and two at the posterior part of the LA. However, several different pulmonary vein anatomy exists [13]. The PVs does not have

valves which hinder reverse flow from the LA. The total stagnation pressure of the blood flow from the PVs into the LA, also called the venous return, is high enough to keep the flow unidirectional.

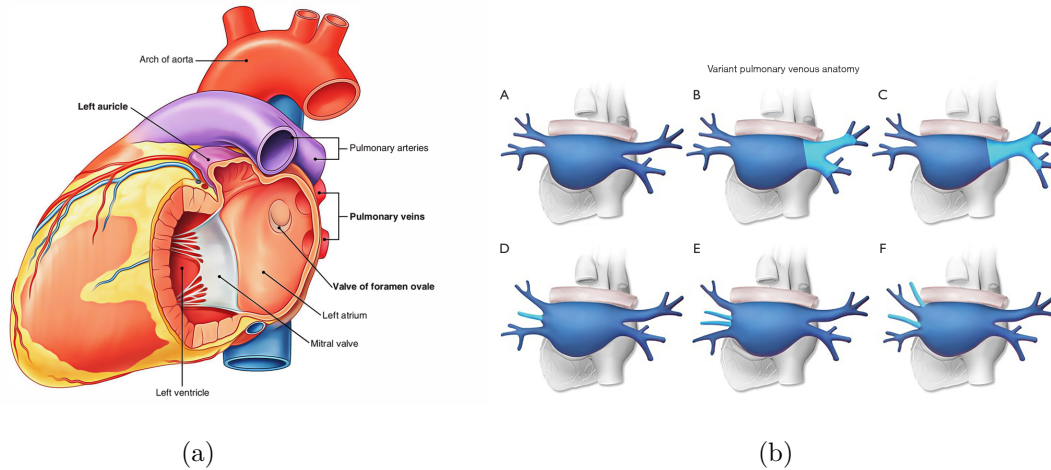


Figure 2.2: Anatomy of the left atrium (a) and examples of varying anatomical differences of pulmonary veins (b). (a) from [14] and (b) from [13].

2.2 Left Ventricle and Aorta

The Left Ventricle (LV) is the driving force of the systemic circulation and has the thickest walls, primarily consisting of muscle tissue called the *myocardium*, in contrast to the LA and rest of the heart anatomy. The LV undergoes large deformation in the cardiac cycle and generates high pressure. Inside the LV are two large *papillary muscles*, which grows into *chordae tendineae*, which again are connected to the mitral valve in order for the mitral valve to not bulge into the atrium when the LV contracts. There are also several irregular protrusions of muscle fibre bundles called the *trabeculae carneae*. Otherwise, the inside of the LV is quite smooth. Blood flows out from the LV to the aorta, which distributes the blood throughout the body. Between the LV and aorta is the aortic valve consisting of three semilunar cusps, which hinders reverse flow into the LV.

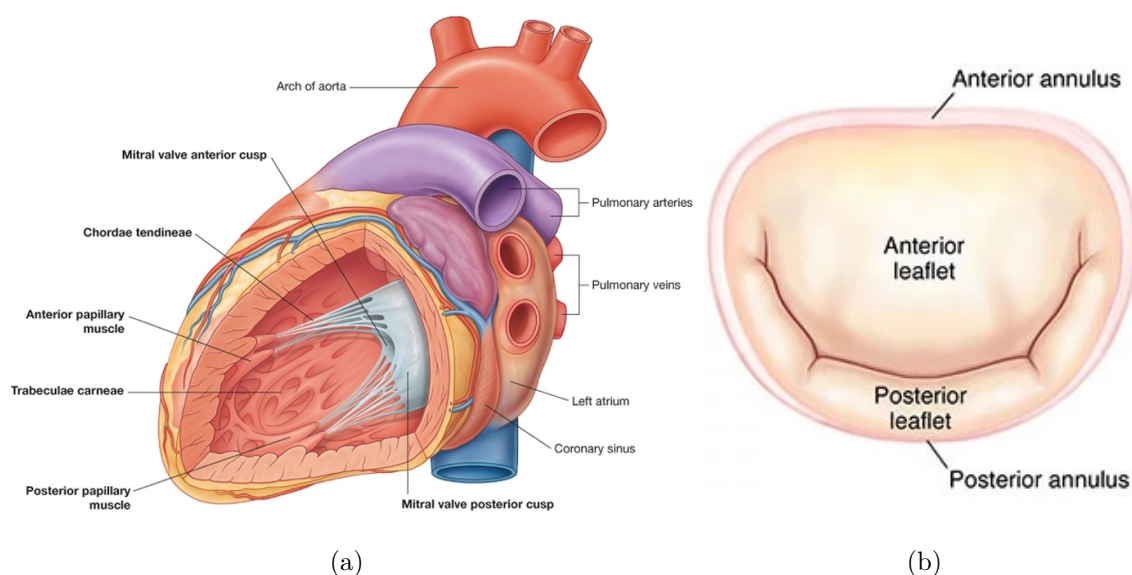


Figure 2.3: Anatomy of the left ventricle and mitral valve. (a) from [14] and (b) from [15].

2.3 Mitral Valve

The Mitral Valve (MV) is connected to a stiff fibrous ring placed between the LA and LV, called the Mitral Annulus (MA). The MV consists of two leaflets, the *anterior* and *posterior leaflets*. The leaflets have large geometrical differences where the anterior occupies a third of the annular circumference and has a rounded free edge. The posterior leaflet is long and narrow and occupies the rest of the circumference. The free edge of the posterior leaflet is often divided into three scallops which gives the leaflet more mobility. The chordae tendineae, which branch from the papillary muscles, are connected to both leaflets. [16]

2.4 Cardiac cycle

The cardiac cycle consists mainly of two primary functions; *systole* where the heart contracts and *diastole* where it expands. These two functions create pressure differences in each chamber of the heart, which determines which valves to open. Both the LV and LA have their own systolic and diastolic cycle, which are quite different from each other. When describing the cardiac cycle, it is most common to base it on the systolic and

diastolic cycles of the LV. Figure 2.4 shows the Wiggers diagram [17], which gives an excellent overview of the volume of the LV along with the pressure of the LV, LA, and aorta. The corresponding Electrocardiogram (ECG) of the electric polarization of the heart is also shown. The cardiac cycle is split up into seven phases and is based on the description by R.E. Klabunde [11].

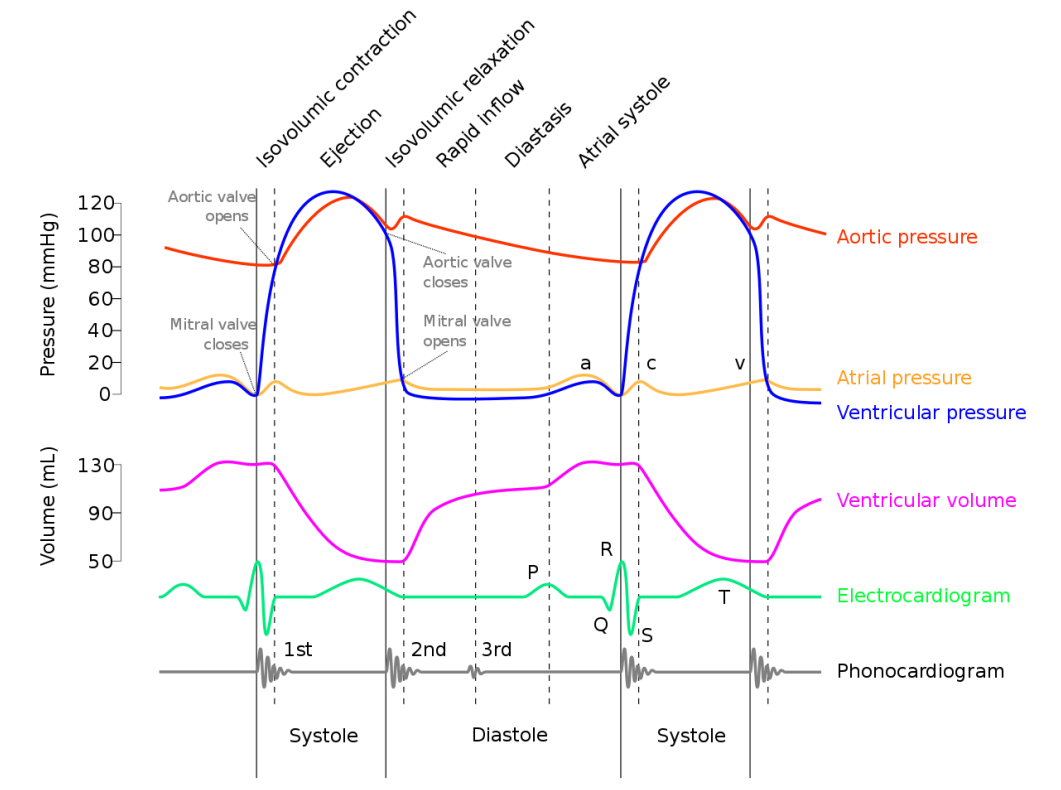


Figure 2.4: Wiggers diagram of the cardiac cycle. Figure from [18].

Isovolumetric contraction: Starting from the QRS-interval in the ECG, the LV has just ended its diastole, and the LV is filled with blood. Both the MV and aortic valve are closed. The systole begins, depolarizing the LV and building up pressure. The blood volume in the LV remains constant, and pressure is built up. In the same time, the LA receives venous return from the PVs and, with bulging of the MV into the LA, P_{LA} rises which is denoted as the c -wave.

Rapid ejection: P_{LV} rises, and when it is above the aortic pressure (P_{AO}), the aortic valve opens, and blood is ejected from the LV into the aorta. The base of the LA is pulled downwards, expanding its volume and pressure is reduced although it is

continuously being filled by venous return.

Reduced ejection: The T-wave in the ECG denotes where the repolarization of the LV begins and starts to relax. P_{LV} drops below P_{AO} , and due to the inertia of the blood, the aortic valve remains open. P_{LA} increases as the LA continues to be filled.

Isovolumetric relaxation: Right before the T-wave ends, the inertial effects decline, and the aortic valve closes. This denotes the end of the LV systole. The LV starts its diastole and begins to expand. The blood volume in the LV is constant, and pressure declines. The LA is still being filled.

Rapid filling: When P_{LV} decline below P_{LA} , the MV opens at the end of the T-wave. Blood is ejected from the LA to the LV; however, P_{LV} continuous to decrease as the LV is still relaxing. Just before the MV opens, P_{LA} reaches its peak denoted as the *v*-wave before it rapidly decreases due to the LA being emptied.

Reduced filling: As the LV continues to be filled and expand, the wall stiffens which causes the pressure to rise and the filling to decrease. This is often called *ventricular diastasis*.

Atrial systole: The LA has thus far filled the LV through passive filling, i.e. the LA has had no muscle contractions. Similar to the QRS-interval for the LV systole, the LA contracts at the P-wave, ejecting as much of the remaining blood in the LA as possible. The contraction causes a jump in P_{LA} which is denoted as the *a*-wave. Inertial effect of the venous return hinders reverse flow in the PVs. After the contraction, P_{LA} decreases until it equals P_{LV} and the MV closes. The diastole ends at the QRS-interval again, where the systole begins anew.

Most of the ventricular filling occurs before the atrium contracts, and is therefore mostly passive and depends on the venous return. This also marks the difference in the systole and diastole of the LA and LV where the LA ejects most of its blood to the LV during the LA diastole. The passive and active filling of the LV is usually denoted as the *E*- and *A*-wave respectively for the transmitral flow. A typical diagram of the volumetric change of the LA can be viewed in figure 2.5 and its characteristic curve is referred to as Sinus Rhythm (SR).

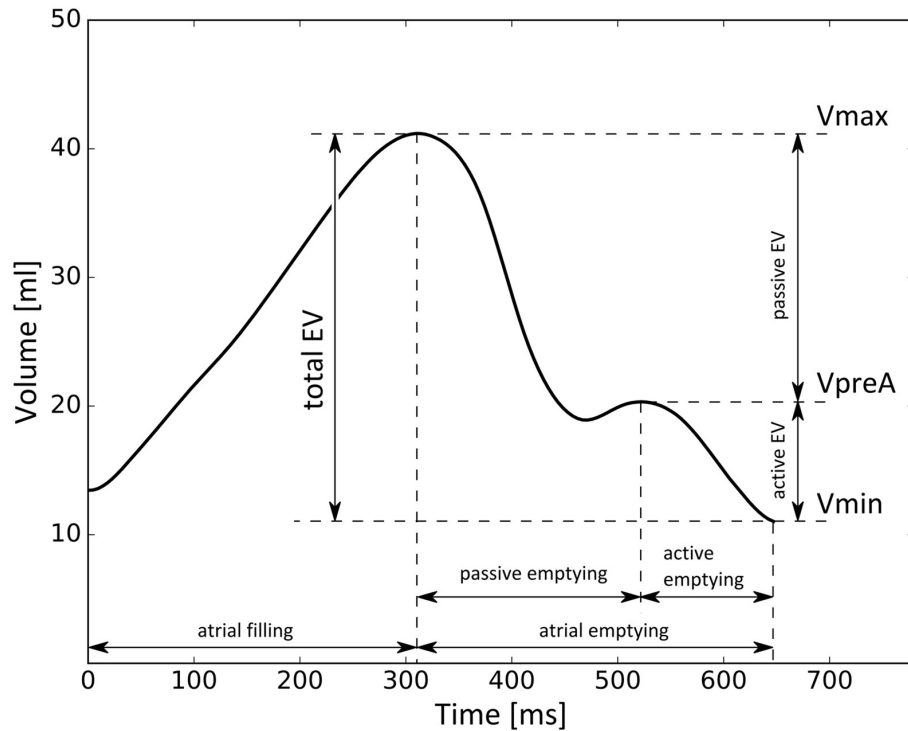


Figure 2.5: Example of LA volume change in the R-R interval of the cardiac cycle. Figure from [19].

2.5 Hemodynamics

Blood is considered as a two-phase liquid consisting of various types of solid cells suspended in liquid plasma. For the fluid behaviour of blood, it is the Red Blood Cells (RBC) which have the most significant impact of its viscosity in larger vessels. The other cells impact the fluid behaviour in the microcirculation where the blood vessels approach the size of the cells. The plasma is viewed as a Newtonian fluid where the viscosity remains constant. RBCs are deformable, making the blood change its viscosity with increasing shear stress and thereby being a non-Newtonian fluid. Blood has a shear-thinning behaviour meaning that with increasing shear-stresses, the viscosity is reduced where the RBCs align themselves along the blood streamlines. Increasing volume percentage of RBC, *hematocrit*, increases both the viscosity magnitude and the shear-thinning behaviour. [20]

Viscosity is determined by the amount of shear stress divided by the shear-rate, deformation caused by the shear-stress. When viewing the viscosity of blood, it is often correlated to the shear-rate. For blood, the shear-stress has to reach a particular value,

yield stress, for the deformation of the RBCs to begin. If the shear stress is below the yield stress, the viscosity goes to infinite, and the blood will move like a solid. Figure 2.6 illustrates a general correlation of viscosity against shear-rate for Newtonian and non-Newtonian fluids and viscosity against hematocrit values.

Hematocrit values also impact the density of blood as the plasma and cells have different densities. Increasing hematocrit will result in an increase in blood density.

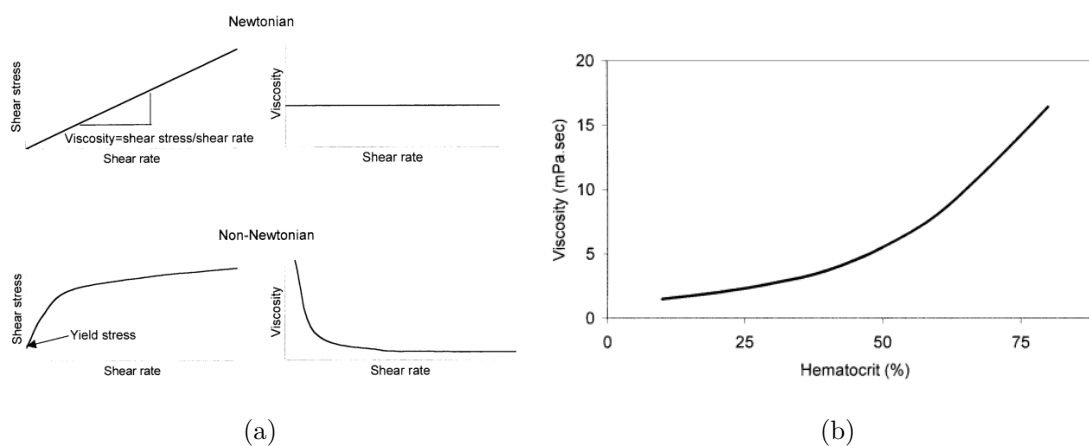


Figure 2.6: Blood viscosity changing according to shear rate (a) and hematocrit values (b). (a) also shows the difference between Newtonian and non-Newtonian fluid property where the non-Newtonian fluid is shear-thinning. Figures from [20].

2.6 Cardiovascular diseases

Presented are a select few CVDs that are of interest regarding CFD-analysis that impact the blood flow in the LA and LV.

Cardiomyopathy is a condition which affects the ventricle muscle making it more difficult to pump blood in the systemic circulation. There are three main types of cardiomyopathy; *hypertrophic*, *dilated* and *restrictive*. Hypertrophic cardiomyopathy involves abnormal thickening of the ventricle muscle, causing narrowing and blockage in the ventricle. Dilated cardiomyopathy involves an enlarged ventricle chamber, making an effective contraction difficult. Restrictive cardiomyopathy involves stiffening of the ventricle muscle, making it less elastic and impacts the movement of the muscle. [21, 22]

Valve diseases are conditions which directly influences the valves in the heart. *Stenosis*

is a condition where the valvular orifice is narrowed caused by thickening of leaflets or cusps being fused together. The narrowing causes a larger pressure difference across the valve, and volumetric flow is limited. *Regurgitation* is a condition where the valves do not close completely, and blood flows backwards. Stenosis and regurgitation can occur with all heart valves. [11]

Arrhythmia is a condition where the ventricle, atria or both have irregular rhythmic cycles. AF, a form of arrhythmia, is of particular interest where the atrium repeatedly beats during the heart cycle and deviates from the normal SR, which impacts the filling of the ventricle. Ventricular arrhythmia is far more dangerous than atrial and usually requires immediate treatment, while atrial arrhythmia has a more long term effect. [23]

Cardiac thrombus is the formation of blood clots in the heart due to blood stasis. Detached thrombi can cause embolism where blood vessels in the circulation get blocked. The thrombus can occur in both LA and LV, usually followed after an infarct, where the heart stops beating, or an arrhythmia. For atrial arrhythmia, the thrombus usually occurs in the LAA. [24, 25]

Chapter 3

Litterature Review

Over the last decade, cardiac computational modelling has come a long way, and it is now common to have a moving LV boundary in CFD simulations. Modelling of the LA and MV does, however, vary in a large degree and strongly impacts the result of the simulations. The accuracy of the model depends on what kind of imaging techniques are used. There is also no consensus on how the hemodynamics shall be treated. This section will therefore review the latest research within cardiac computational modelling to assess how the different modelling impacts the simulation result.

3.1 Cardiac Imaging Techniques

As the heart undergoes large deformations in order to deliver the pressure needed in the systemic cycle, it is a difficult task to model it correctly. There are currently three widely used imaging techniques in order to extract the geometry of the heart; Ultrasound (US) also called *echocardiography*, Computed Tomography (CT) and MRI.

US is the most widely used cardiac imaging technique, especially in the USA, due to its relative low affordability and wide availability. There are, however, several drawbacks regarding US as the probe used to capture images must be placed externally of the human body. This can make it difficult as the anatomy of patients can vary, and certain planes can not be captured. For the left part of the heart, it is the LA and PVs which are difficult to capture. Repeated measurements are also difficult as the captured images are in a considerable degree based on operator skill and machine settings. MRI is able to capture

the entire heart anatomy and is more reliable and reproducible. Drawbacks with MRI are complex scanning protocols where the cardiac cycle is captured in a plane at a time, long scan-time and greater expense and the complications of patients with metallic implants. CT, similar to MRI, is also able to capture the entire heart anatomy. Compared to MRI, it takes far less time to perform the procedure. CT has a higher spatial resolution but lower temporal resolution. CT is also less expensive than MRI making it more available. The largest drawback with CT, however, is the usage of radiation in order to capture the images. However, techniques for lowering the radiation dosage without losing spatial resolution is under development. [26]

US and MRI also have the ability to extract flow measurement. Doppler Velocimetry (DV) is a method used with US that measures flow towards and from the probe. MRI has the possibility to extract a full 3D flow field.

3.2 Left Ventricle

When modelling the inner surface of the LV, *endocardium*, it is most common to neglect the geometrical contribution of the trabeculae carneae, papillary muscles and chordae tendineae. Morud et al. [27] investigated whether the chordae tendineae impacts the intraventricular flow during systole where the LV was modelled from US, and the chordae was modelled using the actuator line method. The results show that at high velocities (above 0.15 m/s), which can occur at late systole, the chordae may cause vortex shedding. The effects are however small, and the conclusion was that the chordae tendineae could be neglected in computational models. Lantz et al. [28] created a highly detailed surface of the endocardium, including the trabeculae carneae and papillary muscles from 4D CT images. The results indicate that trabeculae carneae and papillary muscles strongly interacts with the blood flow, especially when looking at the residence time.

3.3 Left Atrium

Modelling of the LA consists of numerous different variations. The most basic modelling when simulating the LV with the LA is to add a small tubular extrusion above the MA to act as an inlet condition for the MV. Su et al. [29] investigated how a generic LA affects the

intraventricular flow compared to a tubular extrusion. The LV movement was captured from MRI and the generic LA from a CT database. Both versions of LA was static during the simulation. The differences were significant when there was no MV present where vortices from the generic LA was transported into the LV which did not occur with a tubular extrusion. When introducing a simplified MV with prescribed motion, vortices entering the LV from the generic LA was severely reduced but still significant for the intraventricular flow.

When analyzing intra-atrial flow, it is more common to isolate the LA and impose an outlet boundary at the MA. Otani et al. [30] and Masci et al. [31] developed a framework to create a moving LA boundary from CT images, very similar to Lantz et al. [28]. Their objective was to analyze the impact of AF on intra-atrial flow and risk of thrombus. While no definite conclusion could be made, the method to create the LA moving boundary looks promising, and a similar approach is used in this thesis.

Koizumi et al. [32] and Wang et al. [33] also investigated how AF impacts the intra-atrial flow concerning thrombus risk. Koizumi et al. [32] constructed a simplified moving LA boundary from MRI for a healthy specimen and two additional cases were modelled; the first without the final atrial contraction (or kick), which is common during fibrillation, and the second without kick with increased frequency of the general atrial contraction. Results showed that the fibrillation increased the residence time in the LAA, especially in the model with increased atrial contraction frequency. Wang et al. [33] created their own model of how thrombus is formed in the LA. Their result shows that the thrombus aggravates in the LAA during AF and expands into the LA. Their model did, however, not include a moving LA boundary and the AF was imposed as a difference in the mass flow rate at the outlet.

3.4 Pulmonary Veins

How the LA is filled with blood is determined by the PVs. Dahl et al. [34] investigated how different PV locations affect the intra-atrial and transmitral flow. The LA geometry was constructed from MRI and was static during the simulation. MV and LV were not included in the simulation. Three cases were investigated where the PV locations were

placed on the same LA. The inflow of the PVs into the atrium was determined from MRI flow measurements. Results indicate that the intra-atrial flow was strongly influenced by PV locations. The transmitral flow were all skewed with varying degree depending on PV locations. The skewness was in agreement with MRI flow measurement. Conclusions were that patient-specific atria and PVs must be taken into account when assessing intraventricular flow. Lantz et al. [35] also investigated the effect of PV inflow; however, different PV inflow rates were assessed and not location. 20 different PV inflow rates were investigated using a moving LA and LV model from 4D CT images and validated against MRI flow measurements. Results showed that the different flow rates of the PVs affected the intra-atrial flow, however, the regularization effect of the MV nullified these difference in the intraventricular flow. It must be noted that the MV modelling is not clearly described in this article. The MV movement is wholly dependent on the pressure and inertial differences of the LA and LV. Given different intra-atrial flow, the MV movement is also expected to be different.

3.5 Mitral Valve

When assessing the intraventricular flow during diastolic filling, the MV plays a critical role in how the flow develops in the LV. MV modelling is the bane of every cardiac CFD simulations as in contrast to the LV and LA movement, the MV movement depends on the flow inertia and pressure and is not a momentum source. The valve moves depending on the flow and again affects the flow. This interaction between a solid structure and fluids is called Fluid-Structure Interaction (FSI).

The articles presented so far have either neglected the movement of the MV by imposing a simple periodic no-flux condition at the MA or captured the MV movement and applied it in the simulation in the same way the LA and LV as a moving boundary. Mao et al. [36] constructed a 3D FSI model of both MV and aortic valve with moving LV boundary from CT. The results showed some reasonable agreement with US measurement. The model had several simplifications of both the MV strain-energy function and the LA. These simplifications should have an impact on the result. Skallerud et al. [37] constructed a 3D structural model of the MV based on US and found strong evidence

that the valve does not only move due to passive elements but also due to active muscle elements. An accurate FSI model of the MV in the atrioventricular flow thereby becomes extremely difficult.

3.6 Hemodynamics

All cases presented so far have treated the blood as a laminar, incompressible and Newtonian flow with a viscosity of $3.5 \times 10^{-3} \text{ Pa s}$ and density varying between 1050-1080 kg/m^3 . The value of viscosity is generally valid for shear rates above 100 s^{-1} where blood transition from non-Newtonian to Newtonian behaviour. However, during the cardiac cycle, velocity gradients will be reduced or approaching zero, yielding shear rates below this value and the blood will therefore have non-Newtonian properties in the atrioventricular flow.

Doost et al. [38] investigated different hemodynamical non-Newtonian models in order to determine whether the models impacted the simulated results. The simulation was performed on a moving LV boundary from MRI images. Results determined that the non-Newtonian models had a significant impact on the intraventricular flow, especially in the apex region of the LV where there are lower velocities compared to the rest of the LV. A conclusion on which model is best suited, of either Newtonian and non-Newtonian, was not determined.

Under certain times during the cardiac cycle, atrioventricular flow can reach velocities high enough for the blood to transition from laminar to turbulent flow. Chnafa et al. [39] simulated the atrioventricular flow using Large-Eddy Simulation (LES) on a 4D CT image-based moving LA, LV and MV. Results show that turbulence occurs at the MV and apex during diastole and aortic valve during systole. Results were not validated to a laminar solution, and it is thereby uncertain whether large-eddy simulations are more accurate than laminar.

3.7 Discussion

As demonstrated by Lantz et al. [28], 4D CT of the LV and LA results in a more detailed model which can include the trabeculae carneae and papillary muscles due to the high

spatial resolution of CT. In order to include all the details, CT is therefore preferred over MRI and US. The exposed radiation dosage from CT is the only disadvantage, and the opinions are varied of the hazard. MRI is a good alternative if radiation exposure is unwanted. Lacking in both CT and MRI is the modelling of the MV. The MVs that are modelled by CT is extremely simplified. US is far better to capture the MV movement. If a detailed description of the MV movement is wanted, a combined model based on CT or MRI with US is therefore needed.

The LA and PVs are shown to have a significant impact on the MA plane flow. As shown by Su et al. [29], even a static general LA profoundly impacts the intraventricular flow in comparison to a simplified tubular extrusion. Dahl et al. [34] concluded that PV placement impacts the MA plane flow. As the anatomy of the PVs can vary in a large degree between patients, it is therefore necessary to take into account the patient-specific PVs when analyzing patient-specific intra-atrial flow. When assessing the various PV flow rates with moving LA, Lantz et al. [35] showed that it has an impact on the MA plane flow, however, that the MV regularize the differences for the intraventricular flow. A question arises whether it is necessary to take into account a detailed description of the LA movement and PV flow rates when assessing the intraventricular flow.

When assessing how thrombus is formed in the LA, the chances are that they will form in the LAA as demonstrated by Koizumi et al. [32] and Wang et al. [33]. However, as the LAA is able to contract and expand, the static simulations may over predict the blood stasis in the LAA. As such a dynamic case needs to be performed to compare how the LAA movement impacts the blood stasis.

For the atrioventricular flow, the MV is thus far the bottleneck in the cardiac modeling. Any change in the intra-atrial flow is expected to impact the movement of the MV. In 3D models, prescribed motion of the MV is the most used method. Mao et al. [36] have thus far been the only ones able to construct a 3D FSI of the intraventricular flow with the MV and aortic valve. However, due to simplified LA, it can not be determined how different LA impact the FSI movement of the MV. As found by Skallerud et al. [37], the structural modelling of the MV is more advanced than first thought and must be taken into account in order to create a more realistic MV FSI.

The most common hemodynamic model in the simulations is laminar, incompressible

and Newtonian. As shown by Doost et al. [38], non-Newtonian models result in a different solution than Newtonian models. Chnafa et al. [39] also used an LES model and found that several areas with high turbulent kinetic energy. However, with a lack of validation, both non-Newtonian and turbulent models are uncertain to be more accurate or not compared to Newtonian and laminar models.

Chapter 4

Simulation Model Methodology

This chapter will go in-depth on how one can create both a static and dynamic mesh of the inner geometry of the LA from MRI and how it is used in the CFD-software ANSYS® Fluent 19.2 along with how boundary conditions are treated. The MRI has been performed on Rasmus Hvid, a healthy young male, at St. Olavs Hospital, Trondheim 18.09.2019 by Marius Eriksen and is used to generate the geometry of the LA ¹. The LA had four noticeable PVs, which is most common. During the extraction, the heart rate was at an averaged 74.35 beats per minute with standard deviance of 3.13, resulting in a cardiac cycle time of 0.807 s.

4.1 MRI Processing

The MRI images are taken in three different axes of the heart; Short Axis (SAX), Four-Chamber (4CH) and Two-Chamber (2CH). In SAX, the MRI images are taken perpendicular to the longest axis of the heart and give an excellent view of the LV. All four chambers in the heart are visible in 4CH, giving a good view of the LV and LA. For 2CH, the images are placed perpendicular to the MV, giving a good view of the LV and LA. For 4CH and 2CH, the MV is barely visible in certain slices. The SAX images are taken with a resolution of 240x180 voxels of 1.33 mm and spacing 6 mm between the images. 4CH and 2CH have the same resolution with 5 mm spacing. The images have a 12 bit

¹Flow measurements were also performed, however, due to the corona-pandemic, they could not be processed in time for this thesis. MRI of the author was also performed but lacking flow measurements.

depth and 25 frames per cardiac cycle. The cardiac cycle is captured in the R-R interval of the ECG. For a full R-R interval there are in total 26 frames as the first and last frame are the same. The MRI images in the respective axes are shown in figure 4.1.

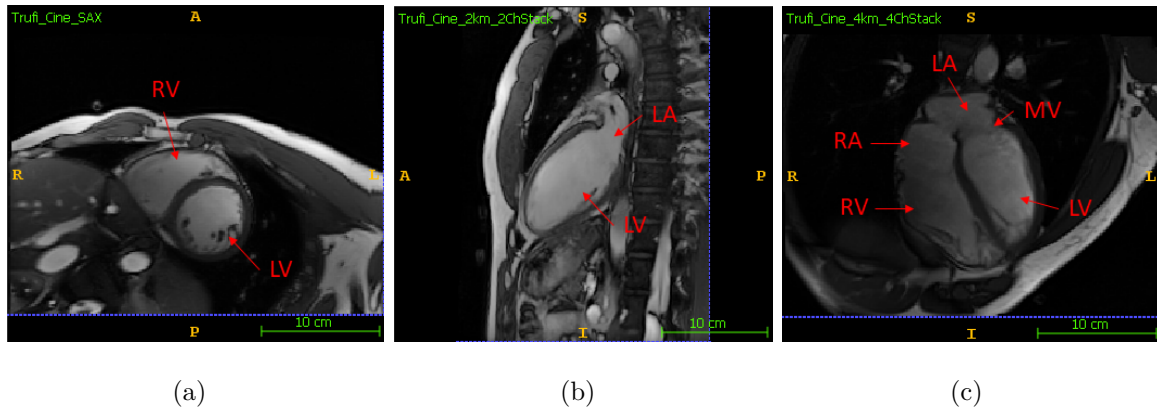


Figure 4.1: MRI of the heart in three different axis. SAX (a), 2CH (b), 4CH (c). Different heart chambers and MV are labeled (Right Atrium (RA), Right Ventricle (RV)).

The cardiac cycle is captured in each plane by averaging multiple cycles and later assembled into a volume in the respective axes and time steps. The operator must place the origin of the plane in reference to each other as best as possible to avoid single planes to be shifted from each other. This is a difficult task, as the respiratory system changes the placement of the heart. During the capture of a heart cycle, the patient holds its breath so that the heart placement is rigid. However, when capturing another cycle, the breath can be held at different lung volumes resulting in a shift of the heart placement. This can potentially yield a large error in the resulting data. For the LV, which has a tubular geometry, post-processing can be done in order to align the planes more correctly. In unpublished work by the author [5], image registration was used to improve the slice alignment. See figure 4.2 for how the registration improves alignment. The LA does not have a similar expected shape to use as benchmark for slice alignment. Post-processing of the LA MRI-slices has therefore not been performed.

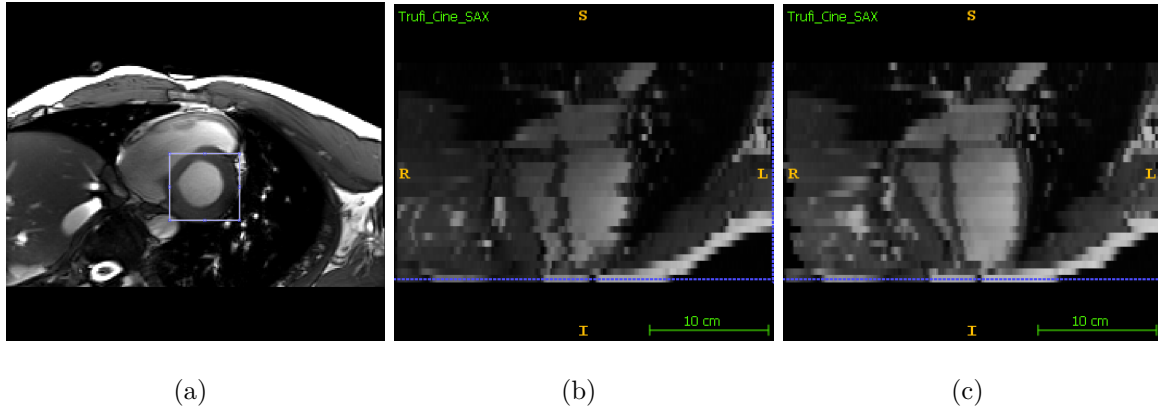


Figure 4.2: Image registration of a SAX series. (a) shows the marking of region to be registered. (b) and (c) show the view perpendicular to SAX axis for pre- and post-registered, respectively.

4.2 Segmentation

The chosen software for segmentation of the LA is Slicer 4.10.2 [40] due to its broad range of functionality and is freely available. A significant challenge when segmenting the LA is the PVs and LAA. The PVs generally have an oval cross-section with diameters spanning from 9.0 mm to 12.1 mm at the *ostia*, the entrance of the PVs to the LA, and decreasing further away from the LA [41]. With slice-thickness of 5 mm to 6 mm of the MRI, the PVs and LAA are poorly represented. The great benefit of Slicer compared to other freely available segmentation software is that different MRI series can be combined into a single volume. Pending on the orientation of the MRI axis, some PV and LAA geometry is captured better than the other axes. By combining the axes to a single volume, the details from each axis are represented. The details of the LA are also increased. As the images may yield errors for each axis, the combination becomes an average of the different axes and reduces the error.

The resolution of the combined volume was not satisfactory, and the resample functionality in Slicer proved useful. With resampling, the spaces between the images were reduced to the highest resolution of the planar images being 1.33 mm before combining the series. The resampling was performed using linear interpolation, which, according to Meijering et al. [42] is best suited for MRI. A consequence when adding the MRI series

together is that the high planar resolution in each series is reduced while the volume resolution is increased. Depending on whether one is only interested in a specific geometry, say the LV, it may be better to only use the SAX series with resampling as details from other planes may not be necessary. Figure 4.3 shows how the resolution of each axis is changed after resampling and combining the volumes.

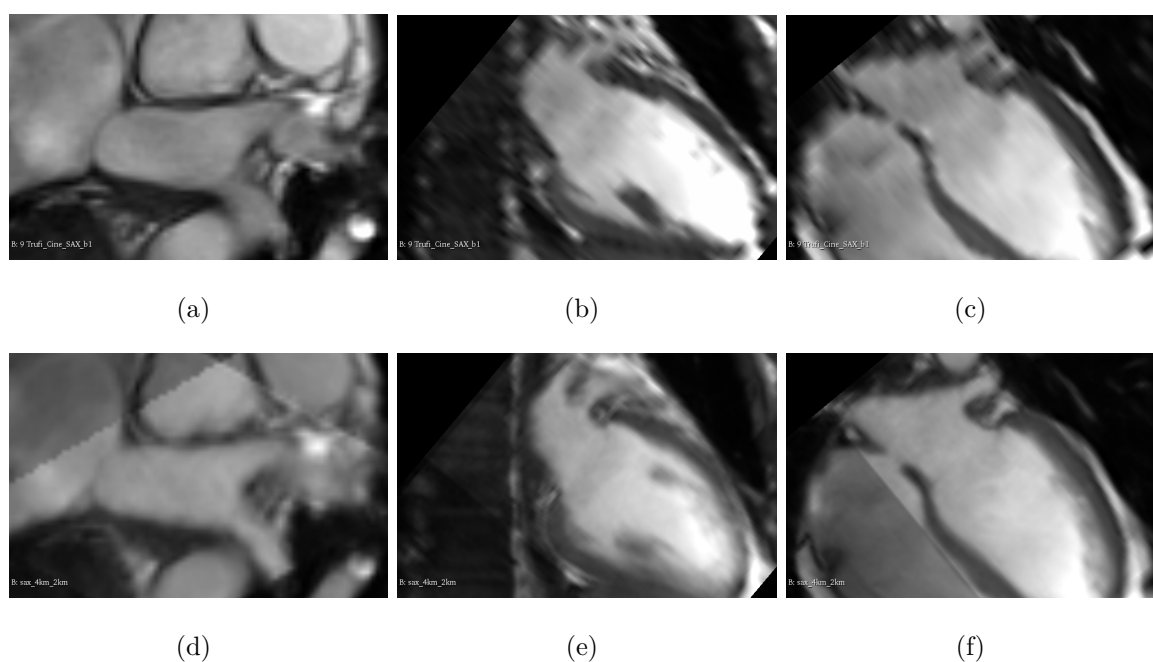


Figure 4.3: Volume resolution before and after resampling and combining axis series. (a)-(c) shows the original SAX volume. (d)-(f) shows the resampled and combined volume of SAX, 2CH and 4CH. It is clear that the in-planar resolution of SAX has been decreased, but the other axes have severely increased.

An important note is that the SAX, 2CH and 4CH axes are not perpendicular to each other. The voxels in the volume are cubic, and there may be some information lost when adjusting the images to the voxel cubes. Taking three MRI series where the axes are perpendicular to each other would probably increase the resolution of the combined volume. The MRI series were taken before knowing that adding the series was a possibility. There was planned to retake the MRI for this thesis with axes perpendicular to each other, however, due to the corona-pandemic, the session was cancelled.

The segmentation is performed by marking the inner geometry of the LA and PVs. Slicer has the functionality to automate the process by finding the inner boundary based

on pixel contrast of a marked volume. With MRI, and especially after combining the volumes, the contrast between the LA and other chambers of the heart becomes blurred. Automating the segmentation process can therefore not be performed. A semi-automated process is instead used where the inner and outer boundary is marked so that the automated segmentation is not able to protrude the marked outer boundary. The inner volume is then automatically segmented. The process is far easier to perform with CT as it has higher resolution and boundaries are clearer.

The most challenging task when performing the segmentation are the PVs and LAA. Even after combining the MRI series, the PVs are still poorly represented. There are other anatomical geometries surrounding the LAA, making it difficult to determine its boundary as the boundaries become blurred in the combined volume. A certain amount of educated guesses is performed in order for the PVs and LAA to look realistic by comparing the segmentation with others such as Lantz et al. [35] and Masci et al. [31]. The PVs were segmented as long as possible, either limited by the MRI-volume or quality.

As the MV is barely visible in certain axes for certain time steps, it was entirely dismissed in the segmentation. The outlet of the LA was chosen at or just below the MA and made planar for outlet conditions in the CFD simulation. The voxels in the volume were oriented after the SAX view, and the outlet plane was oriented after these voxels to make the segmentation easier. Figure 4.4 shows how the MA and a single PV is segmented, along with the complete segmentations.

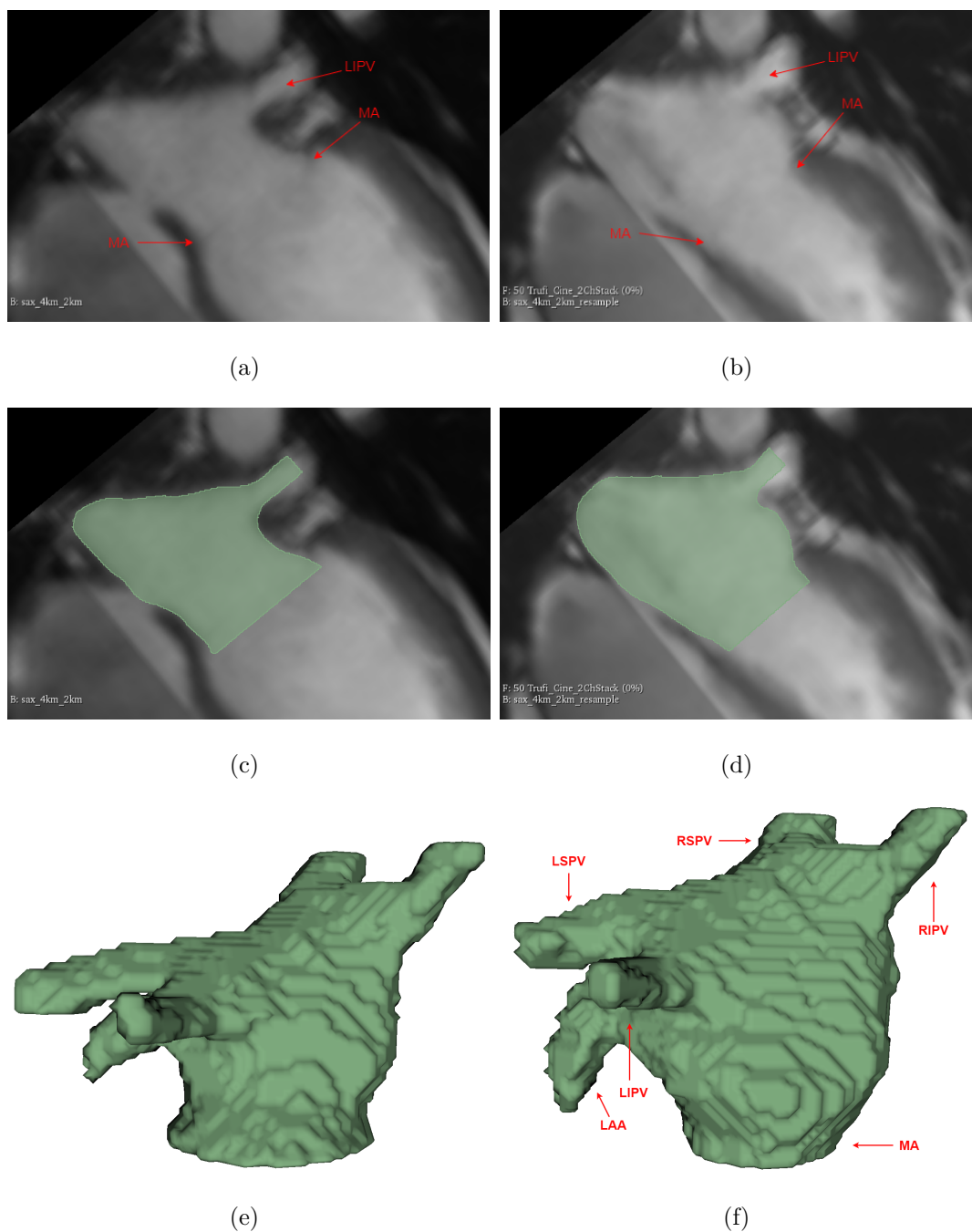


Figure 4.4: Segmentation from MRI. (a) and (b) shows MRI with annotated annulus and a single PV for the 1st and 12th time step respectively. (c) and (d) shows the resulting segmentation in same planar view. (e) and (f) shows the complete segmentation with annotated PVs, LAA and MA. Acronyms of PV labels: Left Superior Pulmonary Vein (LSPV), Left Inferior Pulmonary Vein (LIPV), Right Inferior Pulmonary Vein (RIPV), Right Superior Pulmonary Vein (RSPV). **Note:** Segmentations in (c) and (d) are finished smoothed and cut as explained in chapter 4.3.

All time steps from the MRI were segmented. As all MRI series were taken in the same session, all images are referenced to the same global positional origin. As such the total movement of the LA is captured in the segmentation, both due to its own deformation and translation due to deformation of other heart chambers. The lower part of the LA undergoes the largest amount of deformation while the upper is close to static. This characteristic also helped in segmenting the PVs, where the MRI quality varied between the time steps. The resulting volume of the segmentation can be viewed in figure 4.5 which has a smooth SR. From this diagram, one can also determine where the LV diastole/systole start and ends, which is at the LA maximum volume. This occurs at the 12th time step, which correlates to 0.3551 s after the R-peak of the ECG. This may not be the exact time where the diastole begins but is the best estimate from the temporal resolution of the MRI. The resulting cross-sectional area of the PVs and MA during the heart cycle are shown in table 4.1 which are within reasonable limits of measured cases such as Kim et al. [41] for the PVs, except for the LIPV which may be smaller than it should, and Mihaila et al. [43] for the MA. Note that the naming of PVs are numbered in the numerical treatment of data, as shown in table 4.1.

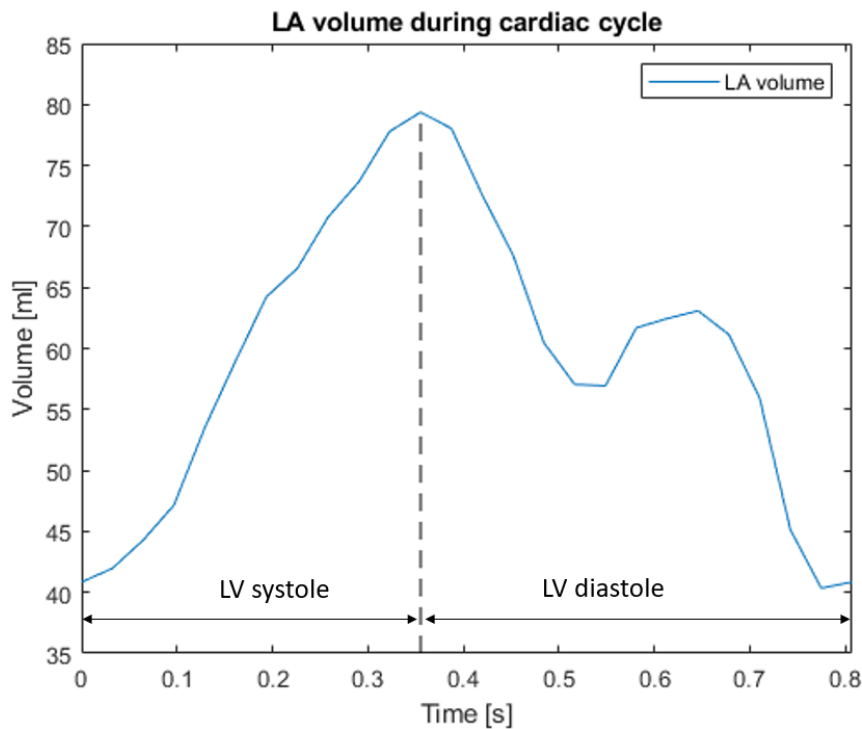


Figure 4.5: Segmented volume of LA. There is great consistency with the LA volume curve from figure 2.5 in chapter 2.4 with a very noticeable SR.

PV/MA	Mean area [mm^2]	Std
LSPV (PV1)	66.2	5.2
LIPV (PV2)	40.8	6.3
RIPV (PV3)	75.9	9.7
RSPV (PV4)	138.1	21.4
MA	813.0	49.3

Table 4.1: PV and MA cross-sectional area.

The segmentations are all slightly smoothed in Slicer to eliminate sharp corners and protrusions. This is important as they will later be used to generate deformation fields in order to create the dynamic mesh.

4.3 Static Mesh

For the static simulations of the LA, two separate time steps were chosen to be analyzed. The first time step is where the LA has its minimum volume, which is at the beginning of the LA systole (R-peak in the ECG) and will be further named T01. The second time step is where the LA has its maximum volume, which is at the transition of LV systole and diastole, further named as T12.

The segmentation from the time steps is exported as Stereolithography (STL) files into the CAD-software ANSYS® SpaceClaim 19.2 in order to further smooth the segmentation. SpaceClaim was chosen as it maintained the coordinate system of Slicer and have great functionality of faceted geometries, and great compatibility with other ANSYS software. The geometry undergoes heavy smoothing as sharp corners are unwanted in CFD-simulations. The overall size of the surface mesh is also chosen in SpaceClaim with the regularize function of the faceted length scale. The initial surface mesh length scale was chosen to be **2 mm** as it managed to capture the geometry of the LA with few sharp corners. After the smoothing, inlets and outlets are hand-cut in the geometry. The PV planes are cut based on what is most orthogonal to the PVs, and the outlet is based on the SAX voxel orientation. The inlets and outlet are then extended in order to eliminate inlet and outlet effects imposed by the CFD software. See chapter 4.8 for how these extensions are determined.

The geometry from SpaceClaim is then imported as a solid into ANSYS® Meshing

19.2 in order to generate the initial mesh. As the LA has a complex geometry, tetrahedral mesh is necessary. The chosen settings were based on trial and error to maximize mesh quality. Five prism layers were added along the LA walls. Three different mesh densities were made for both static cases with mesh length scale of 2 mm, 1.5 mm and 1 mm. Figure 4.6 shows the process of producing the mesh from the segmentation in Slicer. Further information about the mesh settings and statistics can be viewed in Appendix A.1.

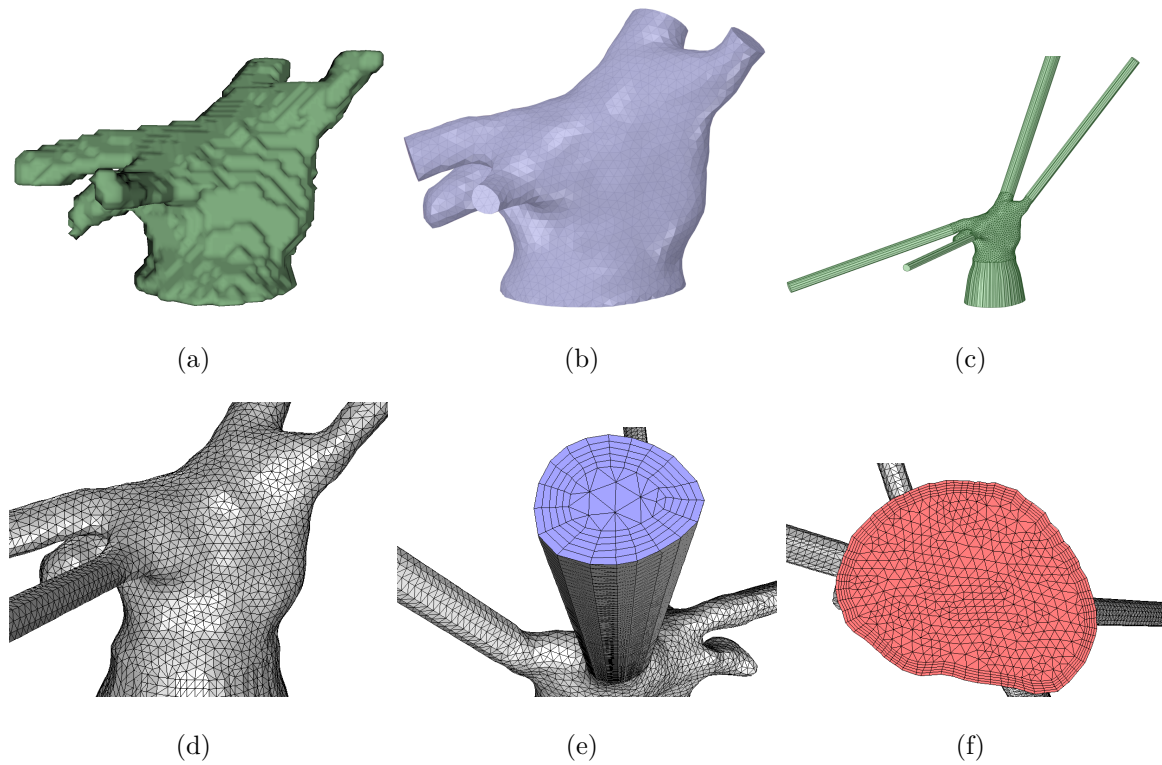


Figure 4.6: Process from segmentation to mesh for T01 with 2mm mesh length scale. Slightly smoothed segmentation from Slicer (a), Smoothed and cut in ANSYS SpaceClaim (b), Added extensions (c), meshed in ANSYS Meshing (d). (e) and (f) inlet and outlet mesh, respectively.

4.4 Dynamic Mesh

The dynamic mesh method is based on prescribed motion of the inner LA wall. The dynamic mesh concept is based on starting with an initial mesh for the first time step and controlling how the surface mesh deforms while allowing Fluent to adjust the internal

mesh. The coordinates of the LA wall nodes will be directly controlled while the inlet and outlet surfaces are determined by plane equations. With this method, it is paramount that the number of wall nodes remains constant throughout the heart cycle. The initial mesh is based on the T01 static mesh.

The deformation field used for transforming the surface mesh is generated in Slicer by use of image registration, specifically *Demon registration (BRAINS)*. The MRI volumes are masked with their respective segmentation where the segmented volume receives a logical value of 1 and the rest a logical 0. In this way, the registration will only take into account the segmented volume. Registration is performed between each successive time step, e.g. 1st to 2nd time step and 2nd to 3rd, as the registration quality is lessened with increasing difference of the volumes. The generated displacement field is shown in figure 4.7. The registration function does have an enormous amount of adjustable parameters. These parameters were slightly experimented with, however, the original settings proved adequate.

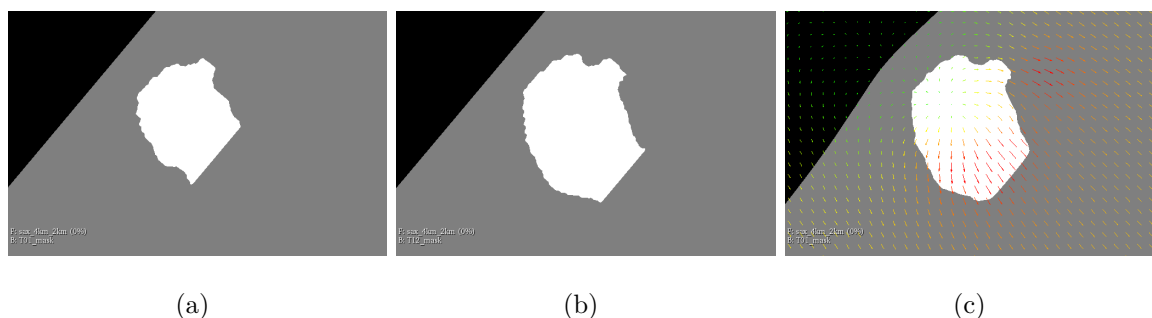


Figure 4.7: Registration of volume between two time steps to extract the displacement field. (a) and (b) are two masked volumes at different time steps. (a) is the volume to be registered towards (b). (c) is the registered volume of (a) with the generated displacement field. **Note:** For illustrating purposes, registration from 1st to 12th time step is shown.

The initial mesh is made in the same way as the static, however without extended inlets and outlets. These extensions were excluded as the nodes on the surfaces were not eligible to be deformed with the deformation field from Slicer, as shown in figure 4.8(a). The extensions could be added after the transformation is performed, however, a small angular displacement at the beginning of the extension becomes a massive displacement at its end and introduce significant momentum sources to the domain, as shown in figure 4.8(b). The

largest deformation of the LA occurs at the outlet and hence became too challenging to add an outlet extension. The mesh file from ANSYS Meshing is imported to SpaceClaim and converted to STL, and then imported to Slicer where it will be transformed between the time steps.

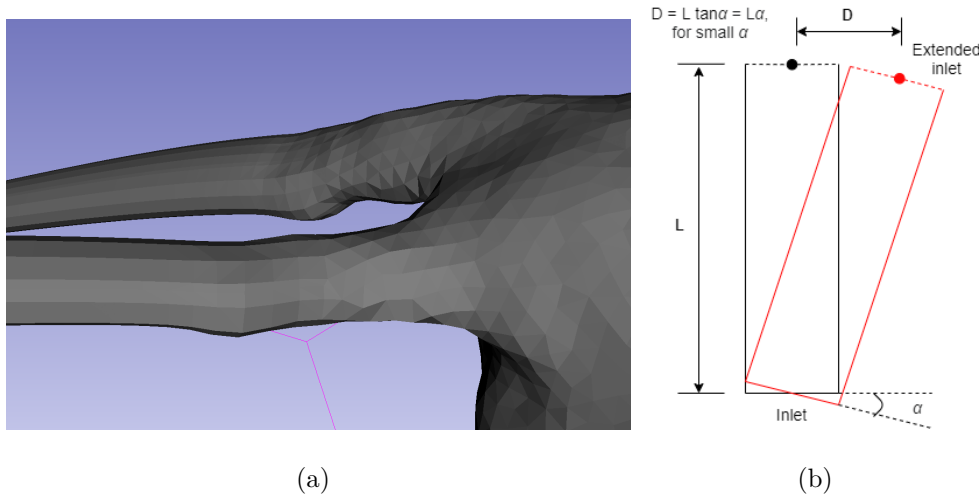


Figure 4.8: Extension errors in dynamic mesh. The initial part of the extensions are unrealistically deformed (a), and displacements become too large (b).

The resulting deformation field from Slicer is then applied on the surface mesh. The transformed surface mesh was not able to keep the inlet and outlet planes planar. Treatment of these planes was performed in MATLAB [44] to make them planar before being transformed to the next time step. Nodes from each plane were identified using the initial surface mesh, and a plane was approximated from the corresponding nodes in the transformed mesh. Using the normal vector and an arbitrary point of this plane, the nodes were moved to the approximated plane along the normal vector by use of the plane equation (4.1a), converted into equation (4.1b). The plane origin was placed at the node with the furthest distance from the approximated plane facing outwards in order for the nodes not to overlap other surface nodes. Figure 4.9(a) illustrates how the nodes are translated. MATLAB script of the plane approximation and node adjustment can be viewed in Appendix C.1. Before the transformed mesh is imported into Slicer for the next transformation, it is imported into SpaceClaim again to adjust geometrical errors which may have occurred from the transformation and plane adjustment.

An unfortunate result of the node adjustments is that the faces in the surface mesh

$$\vec{n} \cdot \vec{QR} = 0 \quad (4.1a)$$

$$Q = P + \vec{n} \cdot (\vec{n} \cdot \vec{RP}) \quad (4.1b)$$

4.1: Plane equation. P is the coordinates of the node to be moved, Q its projection onto the plane defined by the normal vector \vec{n} and R an arbitrary point on the plane.

close to the planes can become highly skewed. This is very visible at the outlet when the LA reaches its maximum volume, as shown in figure 4.9(b). Other nodes could be adjusted to minimise the skewness by the usage of e.g. diffusion equation, however as little adjustments as possible were wanted in order to keep the geometrical integrity.

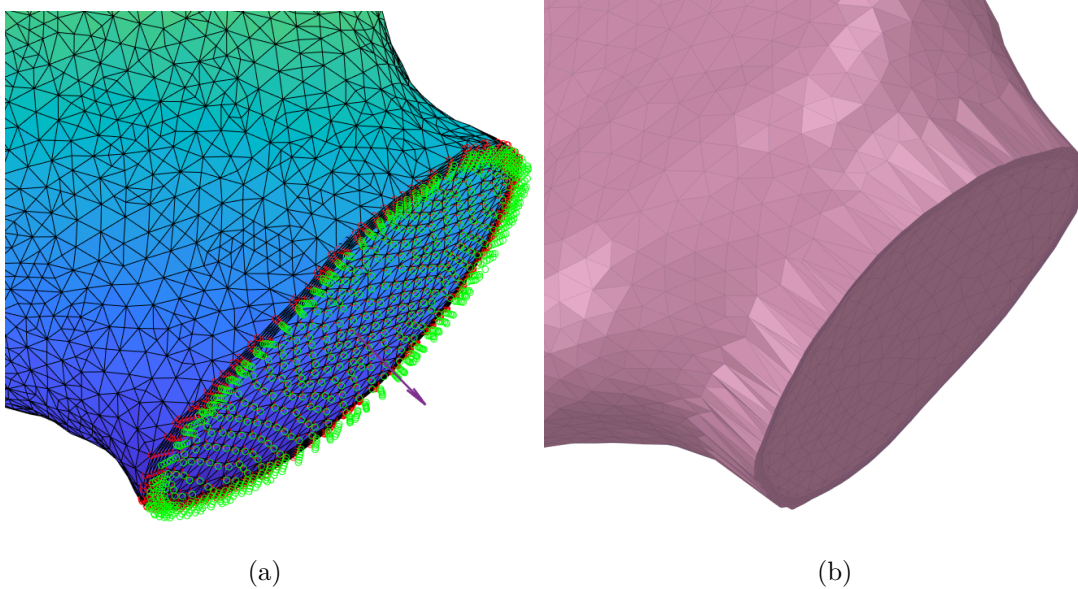


Figure 4.9: Adjustment of plane nodes. (a) shows how the plane nodes are translated from original (red) to new (green) positions using the visible normal vector. (b) shows the maximum skewed surfaces of the 12th time step due to successive node adjustments between time steps. **Note:** The wall nodes of the node adjustment are only relevant as the internal nodes will be smoothed and remeshed, as explained in chapter 4.5. They are however used to approximate the plane.

Transformation of the mesh does have errors which result in relative high geometrical variation in the first and final time step. As the deformation is cyclic, the geometry of the final time step becomes the first again. To address this issue, the transformation is performed in both directions, one forward in time and one backwards, starting from

the initial mesh. The transformed series are joined where the difference in the geometry is smallest, and an average of the node position is made. Unsurprisingly the smallest difference was around the middle of the series, specifically the 12th time step. Figure 4.10 illustrates the transform order.

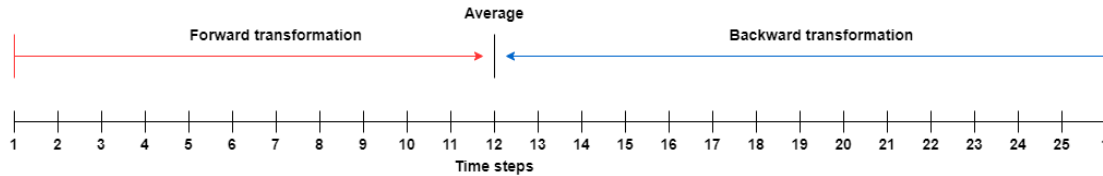


Figure 4.10: Transform order of mesh.

25 STL files have now been created representing the change of the surface mesh of the LA. The current temporal resolution of the dynamic mesh is far too low in order to run a CFD-simulation. Interpolation of the node positions as a function of time is therefore necessary in order to freely control the time steps in the CFD-simulations. Cubic spline interpolation in MATLAB is the chosen method, a 4th order continuous piecewise polynomial function. The method creates a polynomial function for each of the 25 intervals from the STL files representing the cardiac cycle. The polynomials have a continuous second derivative making the spline smooth. Each node is given a spline for each axis in the cartesian coordinate system. The position of every node of the LA wall is thus a function of time. In order to enforce planar outlet and inlets a similar approach is performed. Instead of controlling each node on these surfaces, the plane equation from (4.1b) is used. Splines for the normal vector and arbitrary point for each plane are made in the same way as the wall nodes. The MATLAB script of the spline interpolation can be viewed in Appendix C.2.

An overview of the dynamic surface mesh process using the initial surface mesh and displacement fields is shown in figure 4.11. Two mesh densities for the dynamic case were made using this process with a mesh length scale of 2 mm and 1.5 mm. The length scale of 1 mm was also made but was unable to go through a full cycle in the simulation.

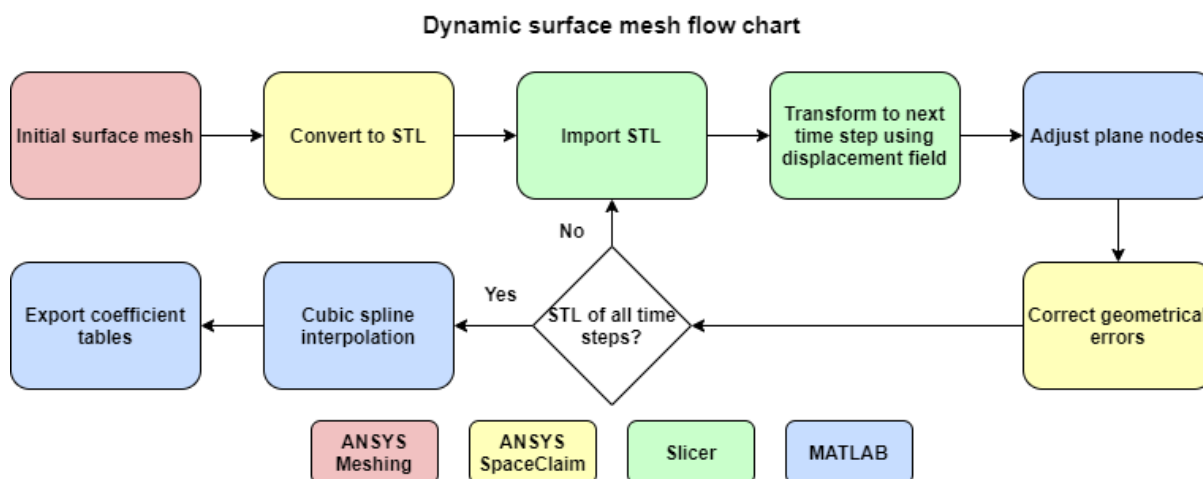


Figure 4.11: Flow chart for generation of time dependent dynamic surface mesh. Corresponding process and software is shown.

4.5 Fluent Dynamic Mesh

Dynamic meshing in Fluent consist of three different methods to control how the mesh deforms; User Defined Function (UDF), Smoothing and Remeshing. There are also other dynamic mesh methods which are not relevant for this thesis.

Smoothing adjusts the mesh of a zone with a moving and/or deforming boundary. As the wall moves in the simulation, the interior nodes are also moved by the selected smoothing method. Mesh deformation is thus distributed throughout the entire mesh domain where smoothing is enabled, which increases the mesh quality. The chosen smoothing method for this thesis is *diffusion*, which uses the diffusion equation to move nodes, as shown in equations (4.2). Smoothing is enabled on the inlet and outlet surfaces and the interior mesh. Other smoothing methods, such as spring and laplacian, were tested but diffusion proved superior.

Remeshing allows Fluent to remesh zones where the mesh quality becomes too poor. Cells are flagged for remeshing if they reach certain criteria which the user specifies. These criteria are minimum and maximum length scales and maximum cell skewness. A cell is only updated if the remeshing improves cell quality. A size function is also used for remeshing where the size distribution of the cells is controlled. In this way, the original size distribution of the cells from the initial time step is maintained. The details of the sizing function are rather advanced and can be reviewed in Fluent user guide [45]. As

$$\nabla \cdot (\gamma \nabla \vec{u}) = 0 \quad (4.2a)$$

$$\gamma = \frac{1}{d^\alpha} \quad (4.2b)$$

4.2: Diffusion equation. \vec{u} describes the mesh displacement velocity and γ the diffusion coefficient. γ in (4.2b) is based on the boundary distance formulation where d is the normalized boundary distance from the nearest boundary and α the diffusion parameter specified in Fluent. α can vary from 0 to 3, where increasing value causes cells further away from the boundary to absorb more of the motion. [45]

such, the inputs of the sizing function was automatically detected by Fluent. Remeshing was enabled on the same zones as the smoothing.

UDFs are functions the user specifies by themselves written in C programming language with inbuilt macros from Fluent. UDFs are applied to the wall, inlet and outlet surfaces of the mesh.

The wall is controlled by the *DEFINE GRID MOTION* macro where the position of every node can be individually controlled for each time step. The polynomial coefficients from the spline interpolation are imported into the UDF where the node positions are calculated from the 4th order polynomial function using the simulation time from Fluent. Every new node position is calculated first in the UDF before it loops over every face and node of the wall to update the node positions. In order to determine which new position corresponds to a selected node from Fluent, a simple tolerance check is used on the difference between old and new position. This method only worked for time steps below 1×10^{-4} s as the difference in the position became too large to identify the nodes correctly. This method was necessary as when simulating in parallel mode, the nodes in each partition changes throughout the simulation. In serial mode, there is only one partition, and an initial nodal identification array could be made as the looping of faces and nodes were performed in the same order for each time step in Fluent.

The inlet and outlet surfaces are controlled by the *DEFINE GEOM* macro where every node in the individual zone is moved by the same function, which in this case is the plane equation from (4.1b). Polynomial coefficients and calculation of plane normal vectors and arbitrary points are performed in the same way as *DEFINE GRID MOTION*.

The planes are using a different macro than the wall as both smoothing and remeshing can be performed on zones defined by *DEFINE GEOM*.

Unfortunately, the planar adjustment of the inlet and outlet surfaces had issues during the dynamic meshing in the simulations where negative cell volumes were generated. A reasonable explanation is that the interpolation of the normal vector did not correspond accurately enough with the wall nodes for the plane zones. The inlet zones caused errors at time steps below 1×10^{-4} s and outlet zone below 5×10^{-5} s. The inlet zones did not deviate that far from the planar geometry as there was minimal PV displacement, and the planar adjustment was therefore voided. The outlet, however, had too large deformation and the plane adjustment was enforced. As such, the minimum time step size for the dynamic mesh was set to 5×10^{-5} s. The maximum deformation of inlet and outlet mesh zones without plane adjustments can be viewed in figure 4.12. A potential workaround would be to calculate the normal vector directly in Fluent for each time step using an UDF, however, this proved too difficult as there was no information to be found on how to identify wall nodes for the inlets and outlet zones in the UDF.

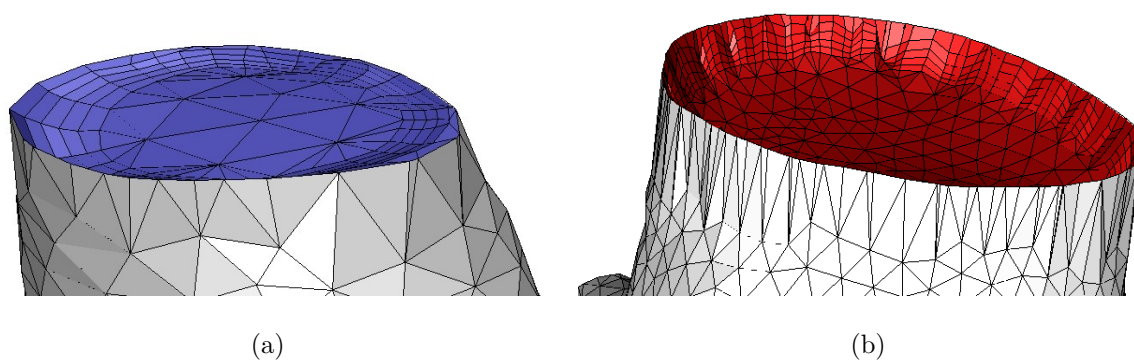


Figure 4.12: Maximum deformed inlet (RSPV) (a) and outlet zone (b).

The mesh quality deteriorates for each completed cycle and eventually crashes. To circumvent this, the initial mesh is reread into Fluent after each completed cycle. The data in the old mesh is interpolated into the new mesh. Final settings for the dynamic mesh can be viewed in Appendix A.2. The UDF for both *DEFINE GRID MOTION* and *DEFINE GEOM* can be viewed in Appendix B.1 with an excerpt of the imported coefficients in Appendix B.2.

4.6 Boundary Conditions and Physics Continuum

The purpose of this thesis is not to perform an exact simulation of the intra-atrial flow, but rather a comparison of the static and dynamic model in order to determine which to use when exact simulations are performed. There is also no alternative to validate the simulation at current time. As such, similar to most of presented literature, the blood is treated as a laminar incompressible Newtonian fluid with dynamic viscosity of $3.5 \times 10^{-3} \text{ Pa s}$ and density 1060 kg/m^3 [20].

Similar to Dahl et al. [34], the boundary conditions are set as transient mass flow inlets for each PV, constant uniform pressure outlet at the MA and no-slip at the wall. With these conditions, the CFD-software will freely determine the pressure at each PV inlet as a reference to the outlet pressure. The outlet flow will be determined by mass conservation. During LV systole, where the MV is closed, there should be no outlet flow, and hence the total PV inlet flow should equal the volume change of the LA. The effects of the MV movement and its no-slip condition are not accounted for in the simulation.

The inlet mass flow is given to Fluent as UDFs using the *F PROFILE* macro. It is possible to use this macro to give the velocity profile a parabolic shape which pipe-like flow has, which is also done by Masci et al. [31]. However, venous and arterial flow has a pulsating behaviour which does not necessarily mimic a parabolic velocity profile [46]. Besides, by using the transformation technique of the mesh in this thesis, the inlet shape is not symmetrically oval or circular, making it difficult to approximate a parabolic function.

4.6.1 Wall Deformation

The deformation of the dynamic mesh does not accurately mimic the local deformation of the LA. The overall deformation may be closely represented, but not how each point on the surface actually moves, as illustrated in figure 4.13. This is important to note as by using no-slip condition, the surface mesh velocity becomes a boundary condition for the fluid at the wall. *MRI tagging* [47] and *Speckle tracking echocardiography* [48] are two different methods able to track specific points of the cardiac tissue which could be used to achieve a specific deformation. These methods are most commonly used to evaluate

the deformation of the myocardium.

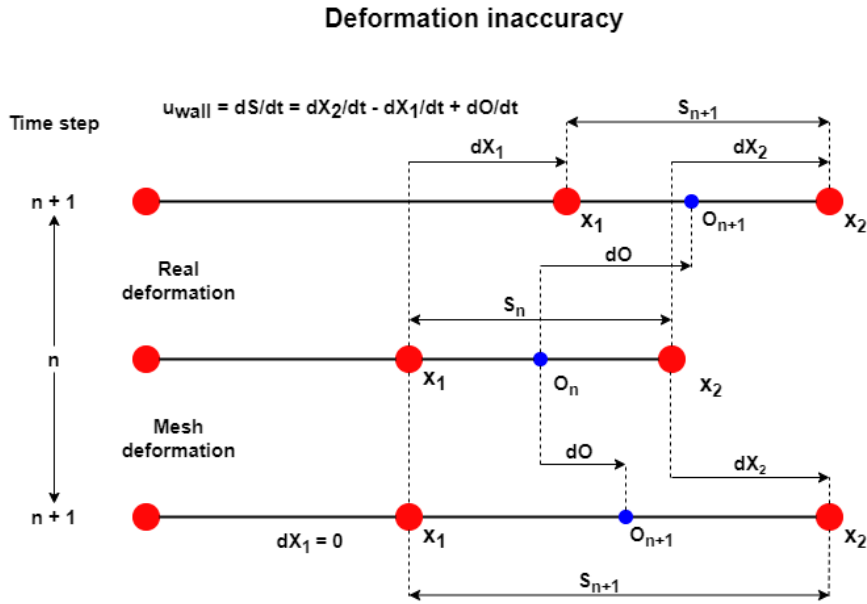


Figure 4.13: Example of deformation inaccuracy which impacts the wall velocity. X_1 and X_2 are points on the surface and defines line segment S with a reference point O at time step n being deformed to time step $n + 1$. Wall velocity, u_{wall} , is defined by $\frac{dS}{dt}$. **Note:** This is a simplified example of how different deformations impacts wall velocity and not how it is actually calculated.

4.6.2 Inlet Condition Dynamic Case

The inlet mass flow is determined from MRI flow measurements from each PV. As mentioned, flow measurements for this thesis were performed but were not able to be processed in time. Instead, the same data from Dahl et al. [34] is used. This will involve potential significant errors in the simulation as the PV inflow and LA volume change originate from two separate patients with different heart cycle times. The raw PV flow data consist of mass flow through the PV orifice. In order to limit the error, the flow data is only directly used during the LV diastole where the MV is open and the LA being emptied. For the LV systole, where the MV is closed, the total mass flow through the PVs are set equal to the volume change of the LA as there should be no mass flux through the outlet. The mass flow through each PV during systole is calculated as a fraction of the volume change where the fraction is calculated from the flow data, as shown in equation (4.3).

$$Q_{PV,i}^{new} = \frac{Q_{PV,i}^{old}}{\sum Q_{PV,i}^{old}} \cdot \rho \frac{dV}{dt} \quad (4.3)$$

4.3: PV mass flow during LV systole. $Q_{PV,i}^{new}$ is the mass flow through each PV, $Q_{PV,i}^{old}$ the mass flow from measured data and $\frac{dV}{dt}$ the volume change from segmentation. **Note:** Only applicable where flow data from each PV are either all positive or negative.

The data from Dahl et al. [34] is split up in its respective systolic and diastolic sections and time adjusted to fit with the new systolic and diastolic sections, as shown in figure 4.14. The data is interpolated using cubic spline interpolation.

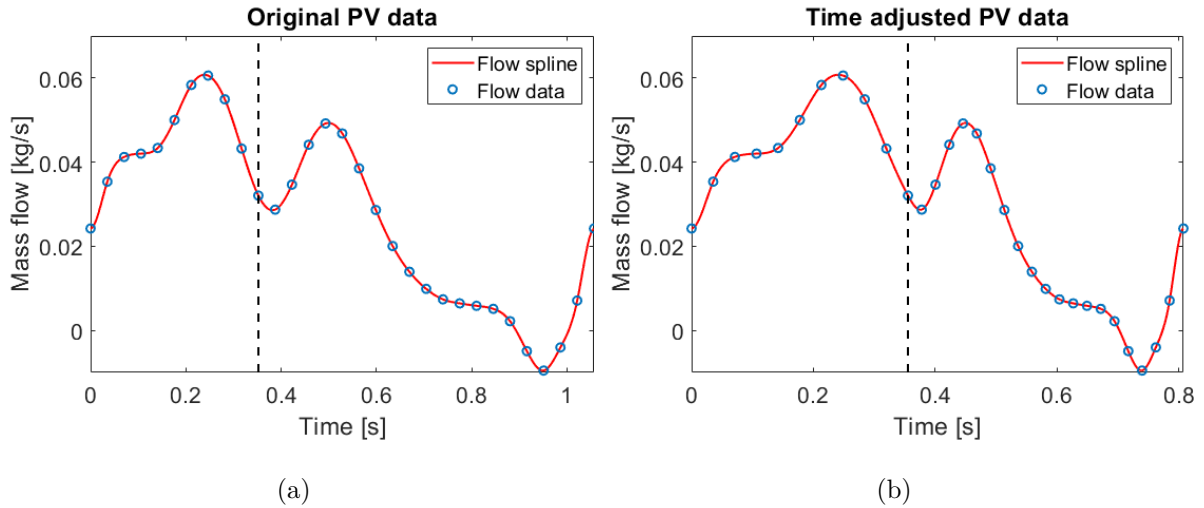


Figure 4.14: Adjustment of PV flow data. (a) is the original data and (b) is time adjusted. Black dotted line marks the split of systole and diastole. **Note:** only a single PV flow is shown.

In order to calculate the volume change, the volume curve from figure 4.5 is interpolated using cubic spline interpolation and differentiated. Resulting curves can be viewed in figure 4.15.

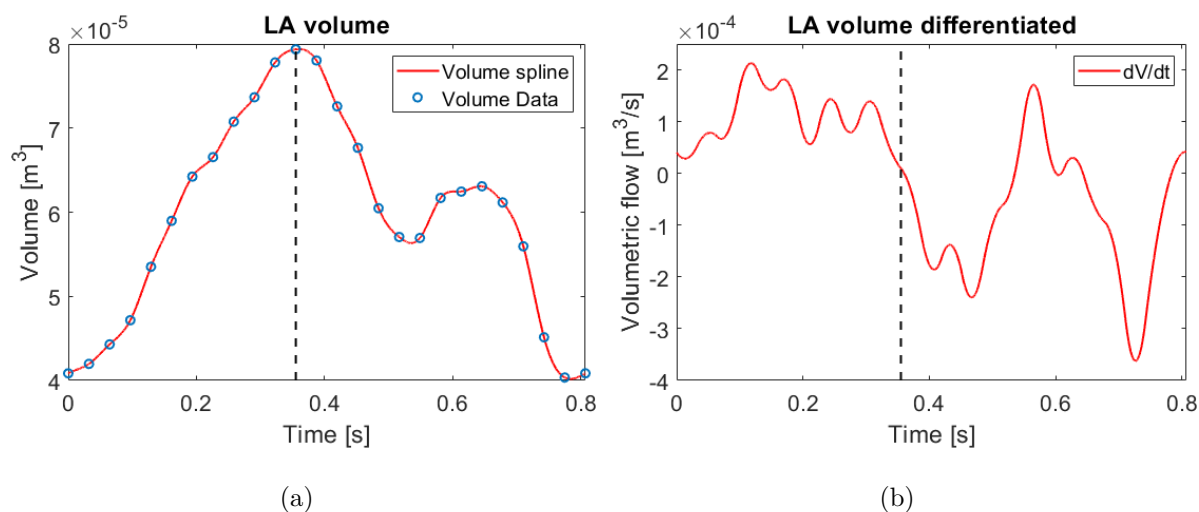


Figure 4.15: Interpolation of volume data and differentiated. (a) is the interpolated volume data and (b) is differentiated. Black dotted line marks the split of systole and diastole.

With equation (4.3), the systolic flow is now modified using the differentiated volume curve. The two curves are non-continuous at their intersection, and modifications are performed to make the flow continuous. Cubic spline interpolation is again used on the intervals between the first data point before and after the intersections. The differentiated values of the curves at these points are used as boundary conditions for the interpolation, and the curve is thus continuous. Figure 4.16 depicts how the new flow curves are updated.

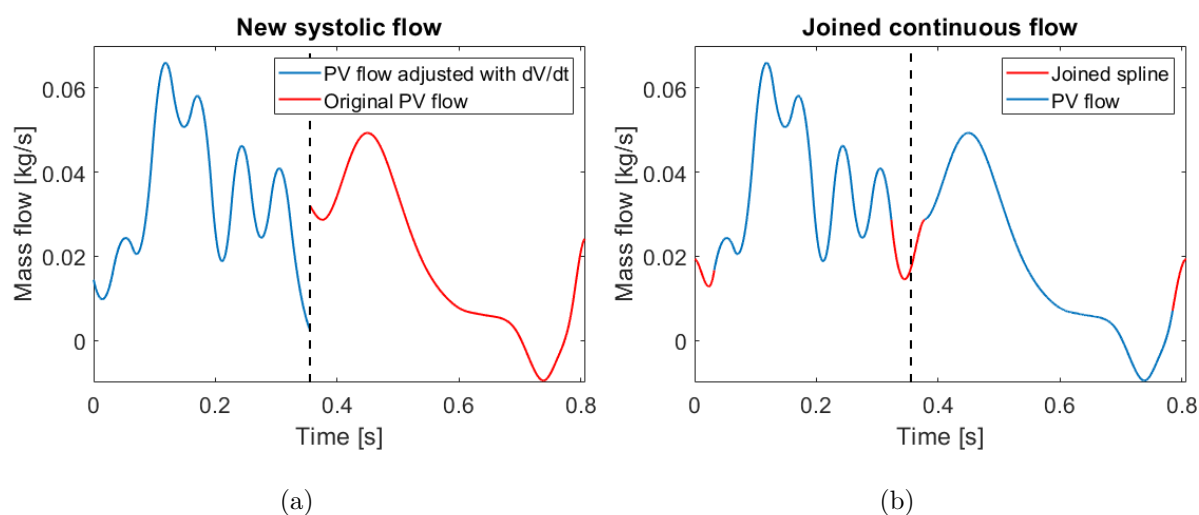


Figure 4.16: New systolic flow. (a) shows the updated systolic flow and is modified to be continuous in (b). Black dotted line marks the split of systole and diastole.

The final result of all PV inflow is shown in figure 4.17 with a comparison of the raw data from Dahl et al. [34]. MATLAB script of the process can be viewed in Appendix C.3 and the UDF in Appendix B.3.

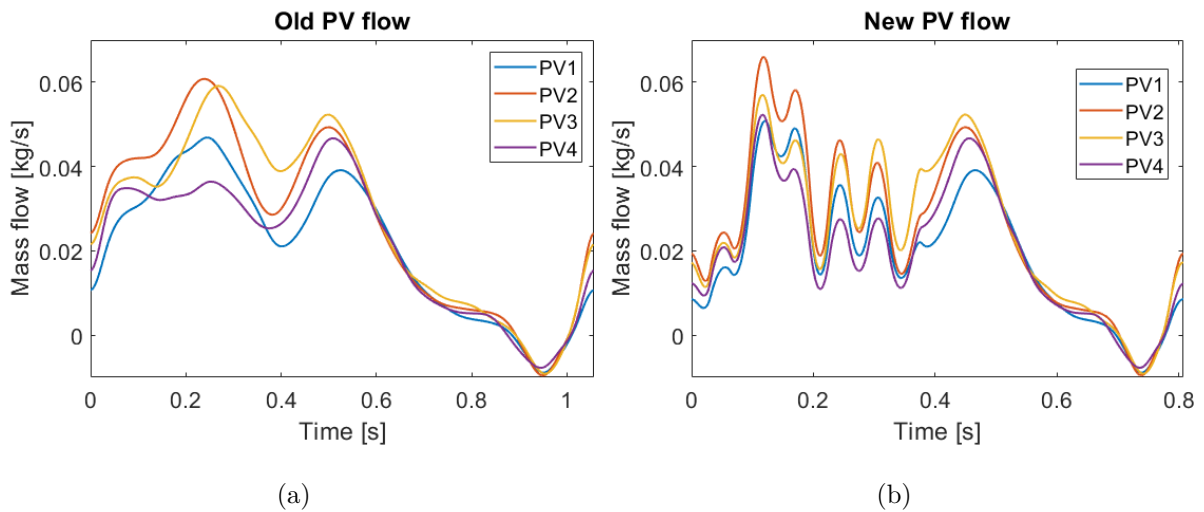


Figure 4.17: Comparison of original and modified PV flow. (a) shows the interpolated original PV flow and the modified in (b).

The expected outlet flow, by imposing conservation of mass, becomes as shown in figure 4.18. During systole, there is no mass flow through the outlet, except for where the intersecting splines were enforced. The E- and A-wave of the transmitral flow is clearly visible as two peaks of the mass flow. It is also notable that the deformation of the LA is the most significant contributor to the transmitral flow.

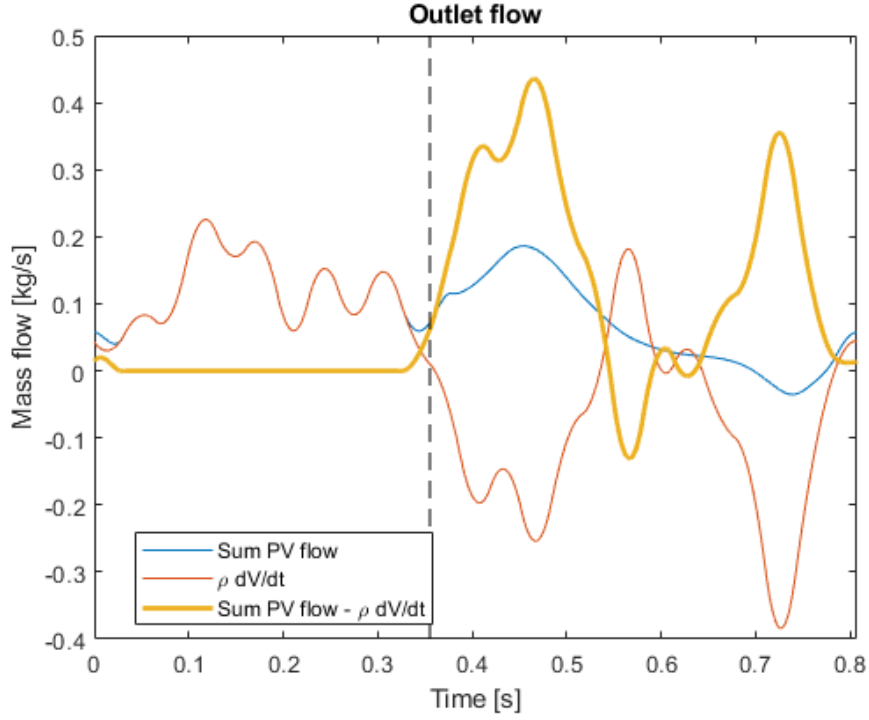


Figure 4.18: Expected outlet flow from mass conservation. Black dotted line marks the split of systole and diastole.

The Reynolds Number (Re) can now be calculated for the inlets and outlet during the full heart cycle to see if the laminar assumption is accurate. Re is calculated as shown in equation (4.4) using the average velocity from the flow curves at the segmented time steps where the area and perimeter of the inlets and outlet are extracted. The result is shown in figure 4.19.

$$Re = \frac{\rho V_{avg} D_h}{\mu} \quad (4.4a)$$

$$D_h = \frac{4A_c}{p} \quad (4.4b)$$

4.4: Reynolds number for pipe flow with near circular cross-section [49]. ρ is the density, V_{avg} the average velocity, μ the dynamic viscosity, D_h the hydraulic diameter, A_c cross-sectional area and p the wetted perimeter.

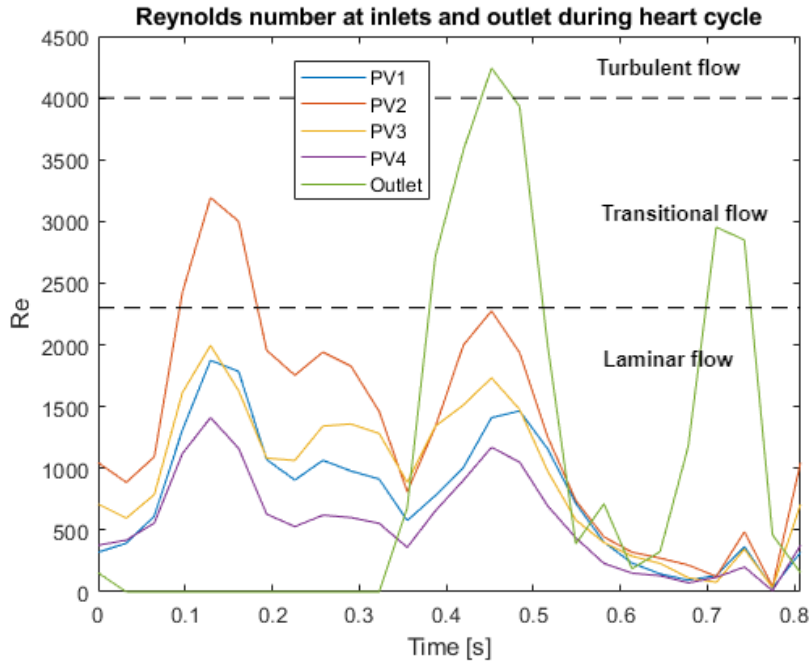


Figure 4.19: Reynolds number for inlets and outlet. Dotted lines mark the region for different flow behaviour. $Re \leq 2300$: Fully laminar flow. $Re \in [2300, 4000]$: Transitional flow, mixture of laminar and turbulent flow. $Re \geq 4000$: Fully turbulent flow. [49]

It is evident that the flow is dominantly laminar for the inlets, except a small spike into the transitional region for LIPV. The outlet does, however, reach full turbulent flow and the laminar model is not applicable for this region where a turbulence model should be used. The turbulent region is however small in comparison to the laminar regions, and a turbulence model may cause more errors for the laminar regions than the laminar in the turbulent regions. As such, the laminar model is kept. For further work, turbulence models for low-Re flows should be investigated, such as the *RANS $k-\omega$ SST* [50].

4.6.3 Inlet Condition Static Case

For the static case, the expected outflow should be the same as the dynamic shown in figure 4.18. However, as the volume remains constant, the inflow is adjusted to compensate for the lack of volume change. The flow curves shown in figure 4.17(b) are used with subtraction of the volume change the same way as depicted in equation (4.3) for each PV, except for the areas in late diastole where the flow fluctuates between negative and positive values. In these areas, the volume change is evenly distributed to all PVs. The

resulting inlet condition for the static case is shown in figure 4.20.

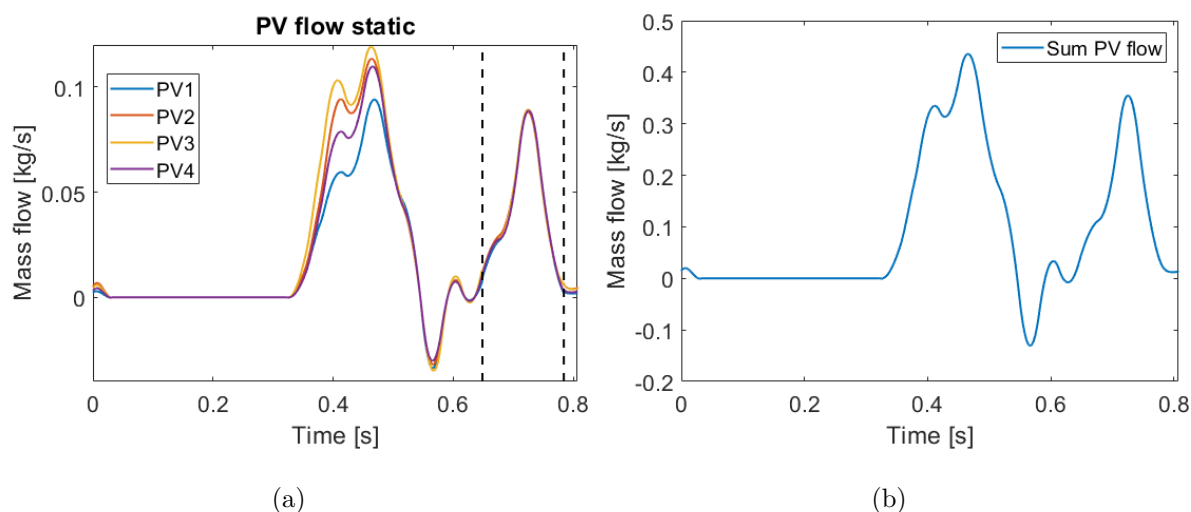


Figure 4.20: Inlet flow condition for the static case. (a) shows the flow for each PV and (b) the total flow. Total flow matches the outlet flow for the dynamic case. The interval of black dotted lines in (a) is where the volume change is evenly distributed among the PVs.

Re is calculated for both static cases throughout the cycle with the area and perimeter taken from the same planes as the dynamic case. The area and perimeter do not, however, vary during the cycle. The result is shown in figure 4.21. Compared to the Re of the dynamic case in figure 4.19, the regions of turbulence are far larger for the inlets. This results from the increased inflow to compensate for the lack of volume change. As the region of laminar flow is still quite large, the laminar assumption is kept. It must be noted that turbulence models should at least be studied for the static cases, but as there is nothing to validate the models to it becomes redundant for this thesis.

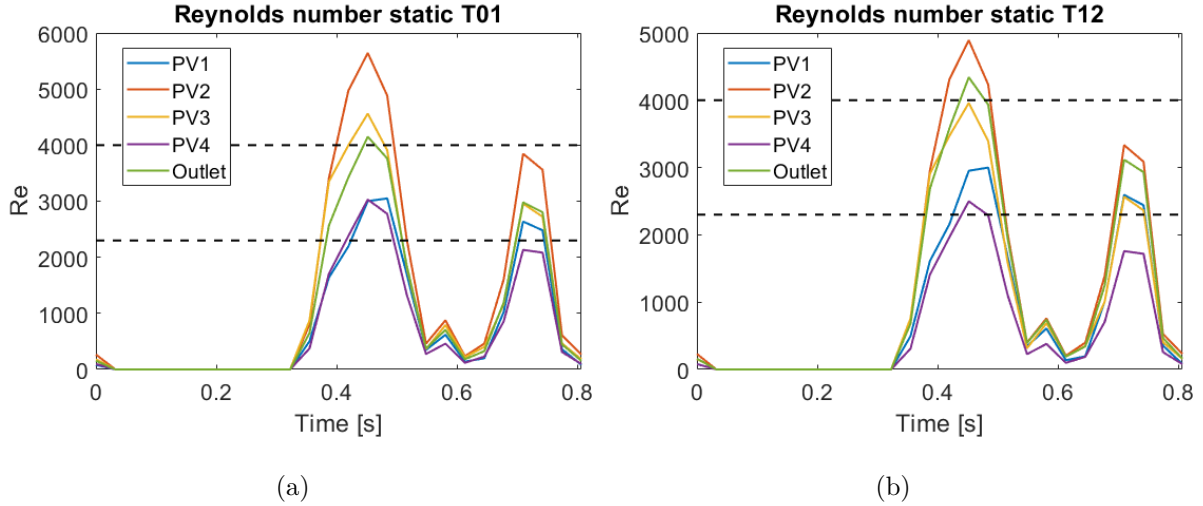


Figure 4.21: Reynolds number for inlets and outlet for static case T01 (a) and static case T12 (b). Dotted lines marks the region for different flow behaviour.

4.7 Fluent Solver

It is critical to choose correct solver settings in a CFD-software in order for the simulation to be stable and give consistent results. Fluent has two fundamental solvers; pressure-based and density-based. The density-based solver is primarily used for supersonic flows, hence the pressure-based solver is used. The following theory is based on *ANSYS Fluent 19.2 Theory Guide* [51] for the pressure-based solver.

4.7.1 Discretisation Theory

For all flows, there are two fundamental equations which describe the motion of viscous fluids based on conservation of mass and momentum. For flows involving heat transfer and compressibility another equation based on conservation of energy is necessary but is not relevant for this thesis. Equations (4.5) shows the conservation equations for a laminar and inertial flow. For incompressible flow, the change in density is neglected.

With respect to dynamic meshes, the integral form of the conservation equation for a general scalar, ϕ , on an arbitrary control volume, V , whose boundary is moving can be written as in equation (4.6). By inserting $\phi = 1$ and $\phi = \vec{u}$ into equation (4.6), the integral form of the conservation equations in (4.5) is obtained.

$$\frac{\partial}{\partial t} \rho + \nabla \cdot (\rho \vec{u}) = 0 \quad (4.5a)$$

$$\frac{\partial}{\partial t} (\rho \vec{u}) + \nabla \cdot (\rho \vec{u} \vec{u}) = -\nabla p + \nabla \cdot (\mu \nabla \vec{u}) \quad (4.5b)$$

4.5: Mass conservation (4.5a). \vec{u} is the flow velocity vector, t the time, ρ the density. Momentum conservation without body forces (4.5b), also called the Navier-Stokes equation. p is the static pressure and μ the fluids dynamic viscosity. Equations in conservative derivative form.

$$\frac{d}{dt} \int_V \rho \phi dV + \int_S \rho \phi (\vec{u} - \vec{u}_m) \cdot d\vec{S} = \int_S \Gamma_\phi \nabla \phi \cdot d\vec{S} + \int_V S_\phi dV \quad (4.6)$$

4.6: S is the boundary of mesh volume V , \vec{u} the flow velocity vector, \vec{u}_m the mesh velocity vector, Γ_ϕ the diffusion coefficient and S_ϕ the source term of ϕ . For stationary mesh, $\vec{u}_m = 0$.

Using the *finite volume method* (FVM), equation (4.6) is applied to each mesh cell and spatially discretised as in equation (4.7).

$$\frac{d}{dt} \int_V \rho \phi dV + \sum_f^{N_f} \rho_f \phi_f (\vec{u}_f - \vec{u}_{m,f}) \cdot \vec{A}_f = \sum_f^{N_f} \Gamma_\phi \nabla \phi_f \cdot \vec{A}_f + S_\phi V \quad (4.7)$$

4.7: N_f is the number of faces enclosing the cell, subscription f denotes value at face f and \vec{A}_f the face area vector. For stationary mesh, $\vec{u}_{m,f} = 0$.

The temporal derivative discretisation of equation (4.6), using first order backwards difference formula can be written as in equation (4.8a).

$$\frac{d}{dt} \int_V \rho \phi dV = \frac{(\rho \phi V)^{n+1} - (\rho \phi V)^n}{\Delta t} \quad (4.8a)$$

$$V^{n+1} = V^n + \frac{dV}{dt} \Delta t \quad , \quad \frac{dV}{dt} = \sum_f^{N_f} \frac{\delta V_f}{\Delta t} \quad (4.8b)$$

4.8: n and $n+1$ is current and next time step, Δt the time step size and δV_f the volume swept out from face f . For stationary mesh, $V^{n+1} = V^n$

In addition to the temporal derivative discretisation, the temporal discretisation scheme

takes into account at which time level the spatial discretisation is assessed. For example, first order explicit and implicit scheme for a scalar ϕ can be written as in equation (4.9).

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = F(\phi^n) \quad (4.9a)$$

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = F(\phi^{n+1}) \quad (4.9b)$$

4.9: Explicit scheme (4.9a) and implicit scheme (4.9b). $F(\phi)$ is the spatial discretisation of ϕ .

Fluent stores the quantity ϕ in each cell centre, and the face values ϕ_f is calculated based on the chosen spatial discretisation method. There are numerous spatial discretisation methods, such as *Upwind*, *QUICK* and *MUSCL*, and will not be gone further into detail. The gradient $\nabla\phi$ also have their own discretisation method. The pressure and velocity are coupled together and can be solved by either segregated or coupled methods.

An important criteria in CFD is the *Courant-Friedrichs-Lewy* (CFL) condition [52], shown in equation (4.10), which applies to the stability and accuracy of transient simulations. The condition is derived from the non-linear convective term, $\nabla \cdot (\rho \vec{u} \vec{u})$, in the Navier-Stokes equation (4.5b) and correlates the mesh and time step size. For instance, if a scalar ϕ is transported with a velocity u in a mesh domain, the time step size must be small enough so that the scalar is not transported beyond the cell size. For simple advection problems, $CFL = 1$ will result in an exact solution. Explicit schemes failing the criteria will be unstable. Implicit schemes are, for the most part, unconditionally stable, but the accuracy will be affected by failing the criteria.

$$CFL = \Delta t \sum_i^n \frac{u_i}{\Delta x_i} \leq 1 \quad (4.10)$$

4.10: CFL condition. CFL , or C_{nr} , is the Courant number. u_i is the flow velocity in the direction of mesh cell size Δx_i and n the number of dimensions.

4.7.2 Fluent Solver Settings

Methods and settings for the solver are chosen to optimize the accuracy and stability of the simulation as recommended by *ANSYS Fluent 19.2 User Guide* [53] for highly skewed mesh such as tetrahedral and deforming mesh. By using the pressure-based solver, only implicit temporal schemes are available.

For the static case, second order implicit scheme is selected as it provides higher accuracy than the first order. The dynamic case uses first order as there will be some discontinuity with the dynamic mesh using higher order implicit schemes. Both static and dynamic cases use the same spatial discretisation methods. For pressure-velocity coupling, the coupled scheme is used as it is superior to segregated schemes for poor quality mesh. For gradients, the least squared cell based method is used as it provides the same accuracy as node based methods but is less computationally expensive for unstructured skewed mesh. The convective term of the governing equations is solved using the third-order MUSCL scheme, which can provide higher accuracy than second order upwind scheme for rotating flows. First order upwind scheme is generally unacceptable.

Unfortunately, the time step size in Fluent is not able to be adaptively adjusted to satisfy the CFL condition. Instead, a fixed time step is used. In order to determine the time step, a single cycle was simulated to evaluate the CFL and adjust the time step. For the static cases a time step size of 1.25×10^{-5} s was sufficient to keep $CFL \leq 1$. For the dynamic cases, the time step size was selected as 5×10^{-5} s, which was the minimum step size before mesh errors occurred.

The solver was set to iterate on each time step until residuals were below 10^{-4} and 10^{-3} for respective continuity equation (4.5a) and momentum equations (4.5b) for the static cases, based on results from outlet extension test in chapter 4.8.2. Residuals of respective 10^{-5} and 10^{-4} for the dynamic cases was set to improve stability. GMRES stabilization method was also applied to the dynamic cases for further stabilization [53]. Each case was simulated through 4 cycles, and data extraction frequency was set to 20 times per cycle. The simulations were run on a home computer with Intel Core i7-8700 CPU @ 3.20 GHz and NTNU high performance computer Vilje using two nodes with Intel Xeon E5-2670 @ 2.6 GHz. ².

²Only two nodes were selected due to long queue times on Vilje.

4.8 Inlet and Outlet Extension

Extensions at the inlet and outlet are added to the static mesh in order to eliminate errors in the simulation caused by the boundary conditions. By extending the outlet, the pressure profile at the MA is able to adjust to the flow. Additionally, reverse flow at the outlet is calculated using stagnation pressure which can involve errors close to the boundary. By extending the outlet, the calculation of reverse flow occurs further away from the LA domain and thus being able to adjust before the flow enters the domain. By extending the inlets, the velocity profile is able to develop before it enters the LA domain and thereby become more realistic in the simulation. In order to determine how long these extensions should be, test cases were performed in Fluent with simplified geometry using the same boundary conditions and solver settings as the primary simulations³. The simulations were performed on a home computer with Intel Core i7-8700 CPU @ 3.20 GHz.

4.8.1 Inlet Extension

For the inlets, a single PV was tested where a simplified geometry was approximated from the first time step segmentation. The selection of the PV was determined by maximum entrance length for the velocity profile to fully develop for a steady-state laminar case with peak velocity from measurements, as shown in equation (4.11).

$$L_{h,laminar} = 0.05ReD_h \quad (4.11)$$

4.11: Entrance length for laminar flow with near circular cross-section [49]. $L_{h,laminar}$ is the entrance length, Re and D_h are defined as in equations (4.4a) and (4.4b) respectively.

The chosen PV to test was RIPV where the entrance length for peak velocity in a steady-state case was 910 mm with Reynolds number of 2450, slightly in the transitional

³The final segmentation for the dynamic case was delayed as it was uncertain which patient to perform the segmentation on due to flow measurement uncertainty. As such, these tests were performed prior to the final segmentation and are based on the segmentation from unpublished work by the author [5] and the raw flow data by Dahl et al. [34] shown in figure 4.17(a) adjusted to 0.807 s cycle time without systole/diastole adjustment.

flow region. The approximated geometry consisted of a straight, circular tube which grows towards the ostium, as shown in figure 4.22(a). This geometry will forwards be referred to as *base* and extended geometry by how far the inlet is extended. There are 6 extended cases with their extension ranging by $5mm \cdot 2^n$, $n = [0, 1, \dots, 5]$.

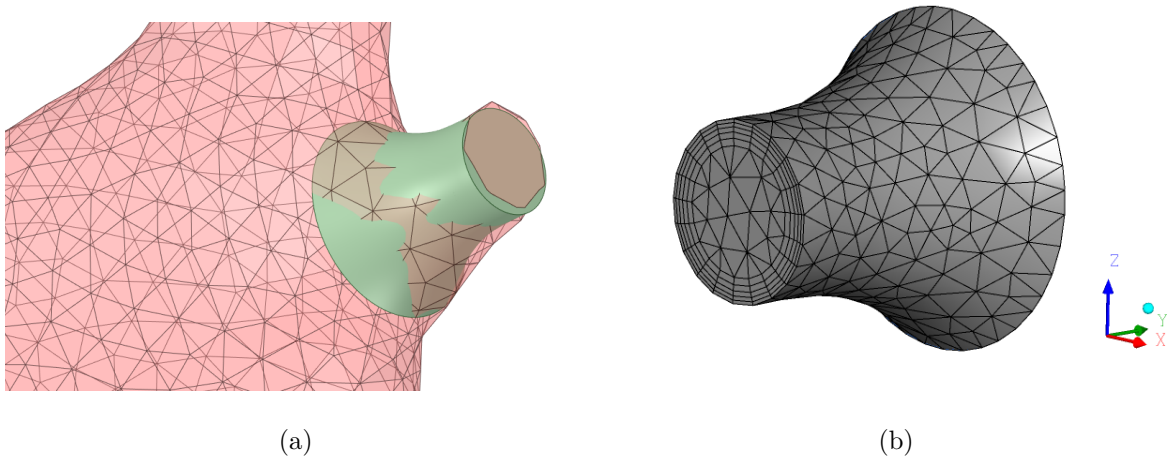


Figure 4.22: Base geometry of inlet boundary condition test (a) and corresponding mesh (b) with axes shown. Red is the LA geometry and green the simplified test geometry.

The geometry is axis-symmetric, and corresponding mesh and solver could be used. However, as the test cases were intended to resemble the main LA simulation, tetrahedral mesh was used with the same settings for the results in the test simulations to be similar to the main simulations. Mesh can be viewed in figure 4.22(b). The solver settings are the same as the main simulations, which is explained in chapter 4.7.2. The time step was adjusted so that $CFL \leq 1$. Residuals were set to 10^{-8} .

Initially, the interior flow is dormant, and the simulation must go through several cycles in order for the interior flow to converge between the cycles. To assess the convergence, velocity in a midsectional plane along the longitudinal axis (YZ/YX plane in figure 4.22(b)) is used. Figure 4.23 shows the Root Mean Square Error (RMSE) of all velocity directions for the base case and **80 mm** extension. It is clear that for the 2nd cycle, subsequent cycles will have, for the most part, $RMSE \leq 10^{-6}$ and considered as converged. Further simulation of inlet extensions will therefore be assessed at the 2nd cycle.

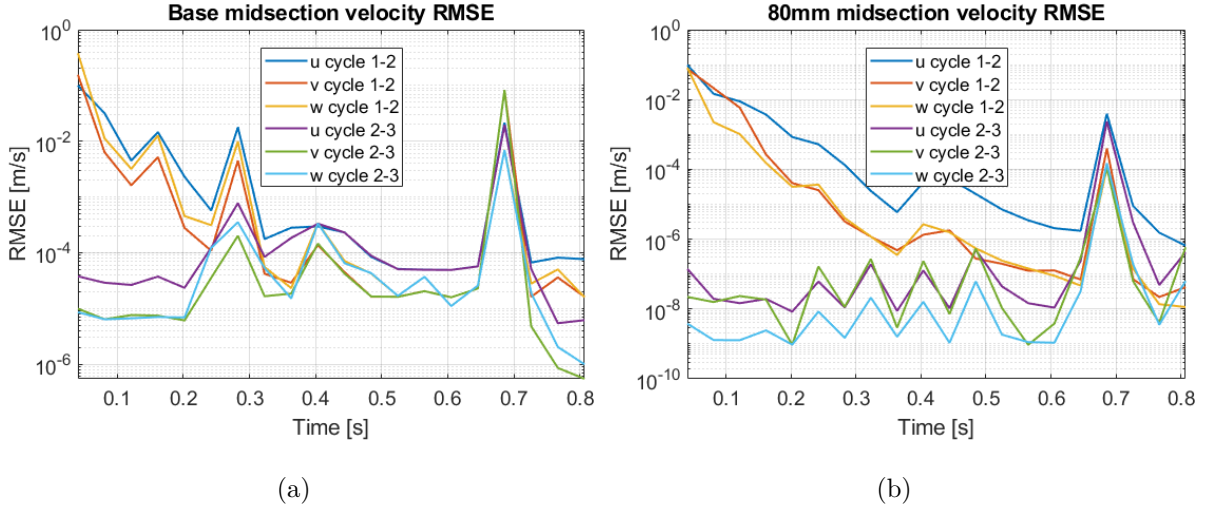


Figure 4.23: RMSE of midsectional velocity components of base (a) and 80 mm extension (b). u , v and w are velocity components of x , y and z axes in figure 4.22(b). RMSE is calculated at each time step between each cycle, 1st to 2nd and 2nd to 3rd.

The spike in the RMSE is due to reverse flow. It is interestingly to see that the RMSE spike does not decrease between the cycles, implying a constant error for reverse flow. Figure 4.24 shows that this error occurs at the inlet and not the outlet, which was expected as reverse flow at the outlet is calculated from stagnation pressure. Additionally, the error impacts the first 10 mm of the inlet length, concluding that in order for the reverse flow to not impact the primary domain, the inlets must be at least extended by 10 mm.

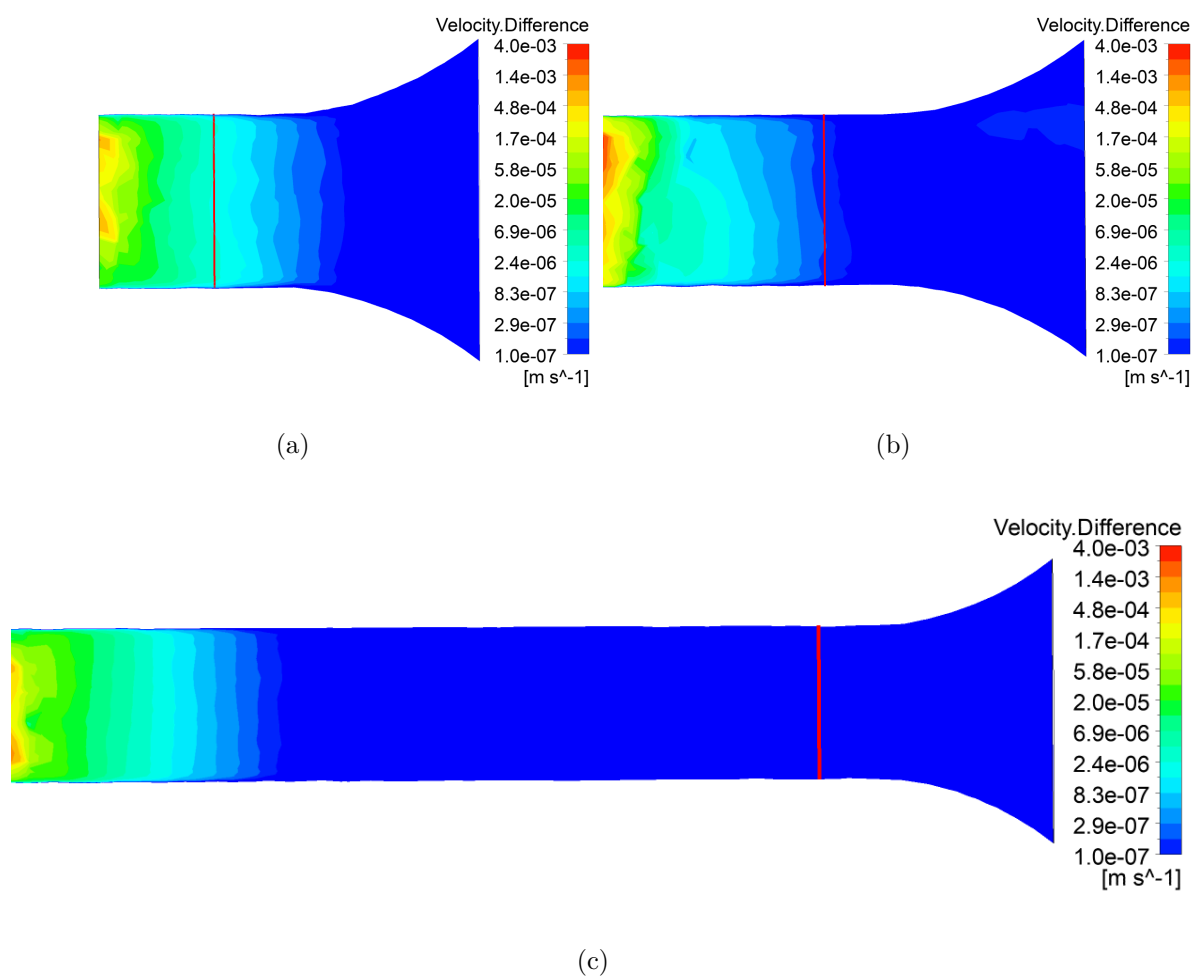


Figure 4.24: Midsectional velocity error during reverse flow at same time step between 2nd and 3rd cycle. 5 mm extension (a), 10 mm extension (b) and 40 mm extension (c). The red vertical line shows the inlet position for base case.

The inlet velocity profile is assessed along a single line at the same position for all cases, located at the inlet of the base case (same as the red vertical line of figure 4.24(b)). Profile at three specific time steps are shown in figure 4.25, which represent the most significant differences between the cases and characteristics of pulsating flow, along with the profile RMSE.

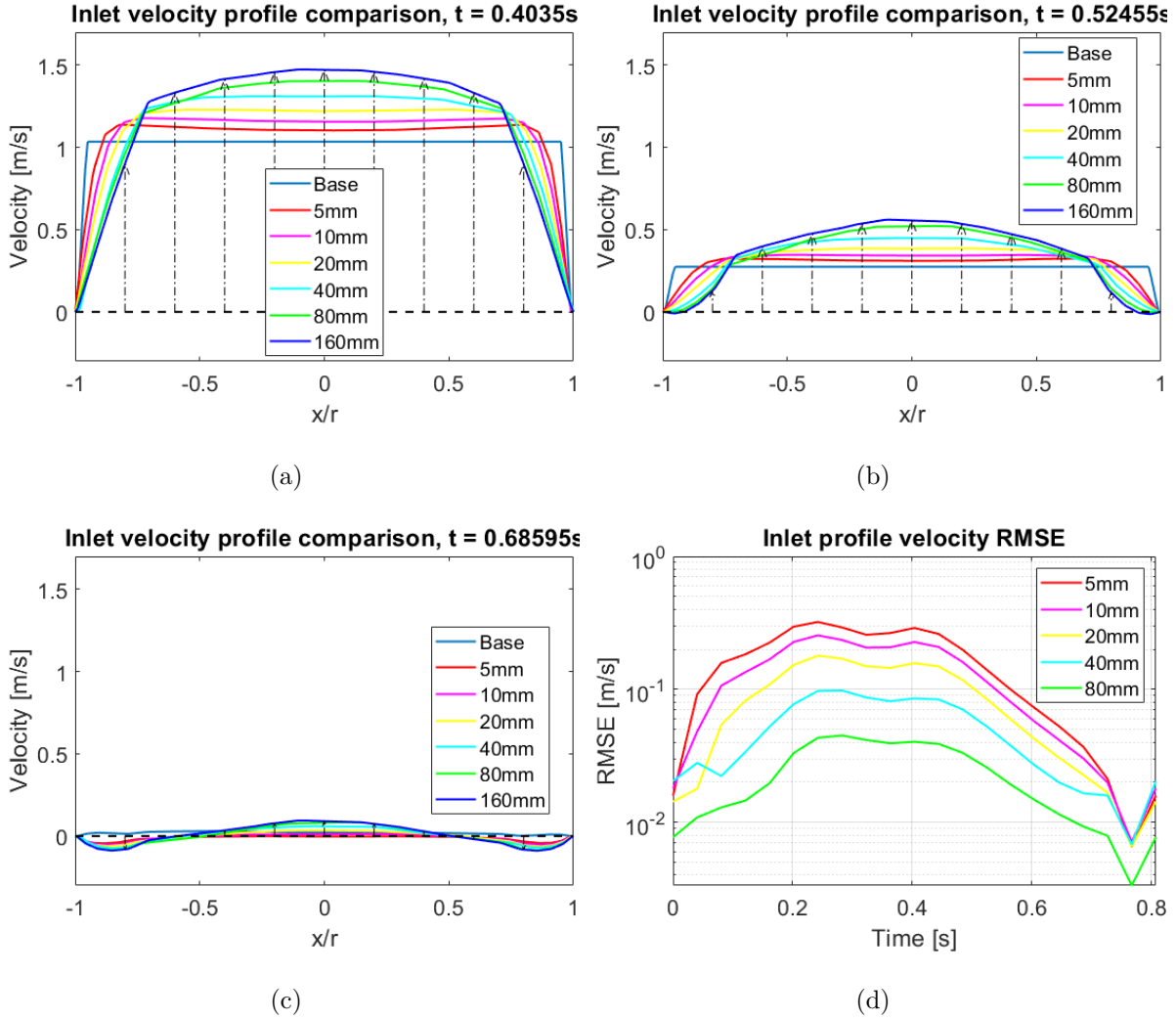


Figure 4.25: Velocity profile for all cases at three different time steps (a)-(c) and RMSE at all time steps (d). RMSE is calculated as difference to 160 mm extension.

It is clear that with extended inlets, the profile is able to develop. With 160 mm extension the profile is quite consistent with profiles from Kim et al. [46]. By dividing mean RMSE from figure 4.25(d) by mean velocity of 160 mm extension, the scaled RMSE is reduced by $\approx 6 \cdot n\%$ from 5 mm and each subsequent $5mm \cdot 2^n$ extension. With a scaled RMSE between 80 mm and 160 mm extension of 3.5%, the velocity profile is considered as converged. Thus, for the static cases in the primary simulation, the inlets are extended by 160 mm. Extended outlet for the inlet test and mesh convergence was also performed where the outlet duct was extended, and cell size was halved with an increasing number of prism layers. Results had minuscule variations and, for the sake of the thesis length, will not be further described.

4.8.2 Outlet Extension

In order to test the outlet extension, a heavily smoothed geometry of the LA was used without the LAA detail. Three test cases were performed; a base where the outlet is placed at the MA, one with a 20 mm duct, and the last with the same duct and an additional 20 mm straight extension. The mesh is made with the same settings as the primary simulations. Geometry and mesh of the cases can be viewed in figure 4.26. Mass flow for each respective PV is set as inlet and uniform pressure at the outlet. Time step was controlled so that $CFL \leq 1$. Residuals were set to 10^{-4} as the simulation struggled to reach lower tolerances.

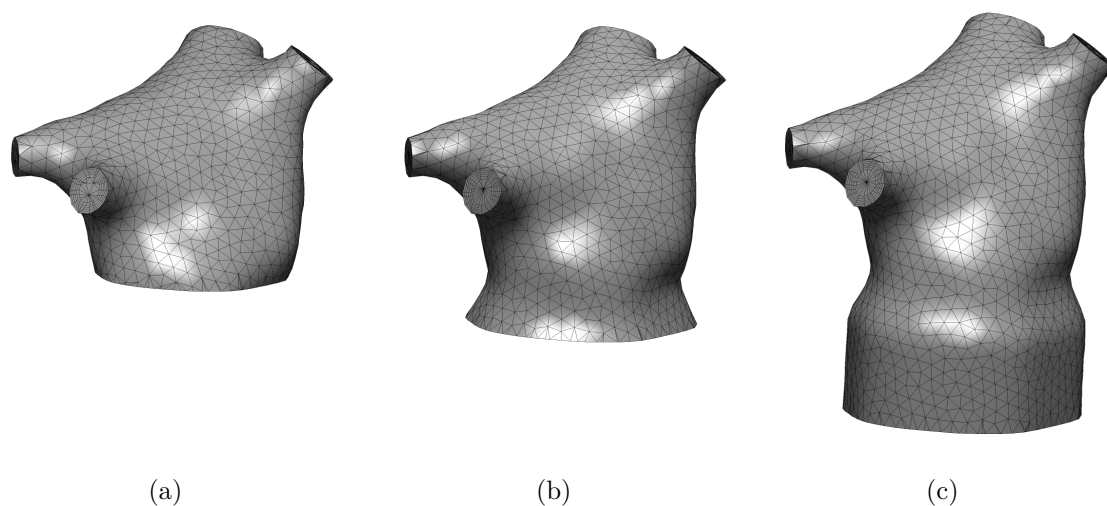


Figure 4.26: Geometry and mesh of the three outlet test cases. Base (a), ducted (b) and extended (c).

Similar to the inlets, the simulation must go through several cycles in order to converge to a solution. Initially, five cycles were estimated to be efficient. The pressure profile of the three cases for a given time step can be viewed in figure 4.28. It is clear that with a duct and extension, the pressure profile is not uniform.

However, the solution does not converge. Figure 4.28(a) shows the RMSE of the pressure profile for the extended case, which indicates that the solution does not tend to convergence during the five cycles. An additional three cycles were simulated, and the velocity RMSE in the volume is shown in figure 4.28(c) with no improvement to the convergence. To check if the solution is stable, i.e. that there is no momentum build up

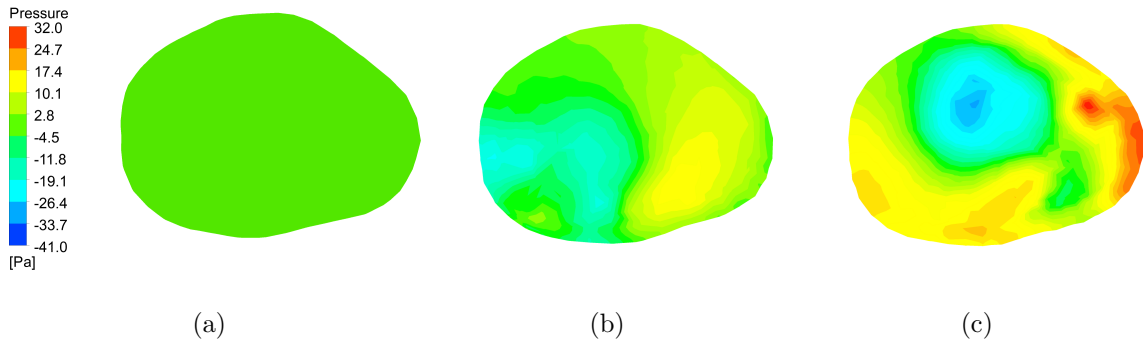


Figure 4.27: Pressure profile at the MA. Base (a), ducted (b) and extended (c).

after each cycle, the kinetic energy, calculated from volume integral of dynamic pressure, is assessed for each cycle. Figure 4.28(b) shows the difference of the kinetic energy, which is quite stable around 10^{-5} . All of the cases had similar tendencies. The solution convergence is therefore not an issue of simulation stability, but rather due to the non-linearity of the Navier-Stokes equation (4.5b). Compared to the inlet the tests, the outlet tests have far more complex geometry and flow conditions, making a highly accurate cyclic result difficult.

As such, there is no clear conclusion to be drawn for the extended outlet. The outlet for the static cases in the primary simulations was chosen to have a **20 mm** duct and **20 mm** extension. In this way, the MA pressure profile should at least not be impacted by the outlet boundary condition. Inlet extensions and finer mesh was also simulated but gave the same tendencies for convergence. A conclusion to note, however, is that all variables had 10% to 20% scaled RMSE, meaning that the residual limit can be raised as it is not the limiting factor of the errors.

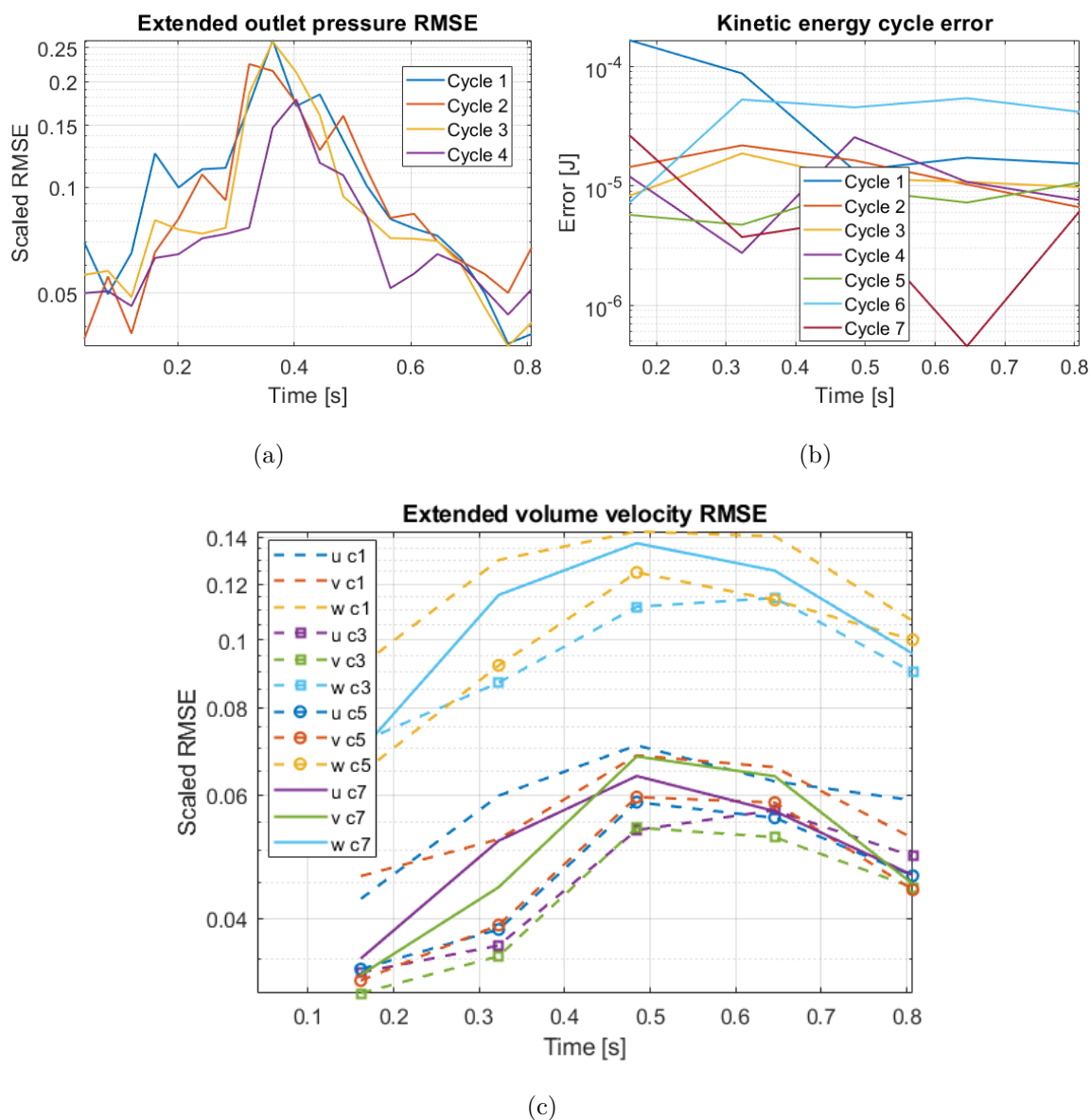


Figure 4.28: Error plot for the outlet test case. (a) shows the RMSE of the pressure profile of MA for the extended case, compared to cycle five and scaled against its max value. (b) shows the kinetic energy error compared to cycle eight for the extended case. (c) shows the RMSE of velocity components of the volume for the extended case, compared to cycle eight and scaled against its max value. Cycles are abbreviated as e.g. c1 = cycle 1.

Chapter 5

Simulation results and discussion

All of the planned simulations were not able to be simulated in time. The finest meshes of the static cases took far longer time to simulate on the selected number of nodes on Vilje and were only able to complete three cycles. The dynamic cases received arbitrary errors on Vilje (Linux OS) which did not occur on a home computer (Windows OS). There was only time to simulate one of the dynamic cases which were selected to be the coarsest mesh. Results will, therefore, be presented for the selected cases: Static; T01 and T12, 2mm and 1.5mm mesh scale four cycles, 1mm mesh scale three cycles. Dynamic; 2mm mesh scale four cycles.

In order to identify characteristics of the intra-atrial flow, three flow phenomenons are evaluated between the different cases; streamlines, vortex structures and wall shear stress.

Streamlines indicate the instantaneous direction of fluid motion throughout the flow domain. The streamlines are defined as tangent lines to the velocity direction, which can be expressed as equation (5.1).

$$\frac{dx}{u} = \frac{dy}{v} = \frac{dz}{w} \quad (5.1)$$

5.1: Equation of streamlines for a cartesian coordinate system [49]. (x,y,z) are spatial directions and (u,v,w) the corresponding velocity components.

Vortex structures visualise the vorticity in the flow domain. Even though there is low transmitral flow, vorticity may increase the local intra-atrial flow velocity and prevent

blood stasis which again prevents thrombus formation [31]. The chosen method to visualise the vorticity is the Q-criterion, the second invariant of the velocity tensor, as explained in equation (5.2).

$$Q = \frac{1}{2}(\|\Omega\|^2 - \|S\|^2) \quad (5.2a)$$

$$\Omega = \frac{1}{2}(\nabla\vec{u} - \nabla\vec{u}^T) \quad , \quad S = \frac{1}{2}(\nabla\vec{u} + \nabla\vec{u}^T) \quad (5.2b)$$

5.2: Q-criterion of vorticity [54]. Ω is the vorticity tensor and S the rate of strain of a fluid element. $Q > 0$ defines where the vorticity tensor is greater than the rate of strain.

Wall Shear Stress (WSS) can be used as an indicator of CVD, correlating blood flow and tissue behaviour. Low WSS indicates high residence time of blood which can form thrombus for residence time lasting 10 s, while high WSS can cause tissue ruptures and thrombus, however, by a different biomechanical method than low WSS [55]. WSS is defined as in equation (5.3). Low WSS threshold occurs for shear rates below 100 s^{-1} , which lies in the non-Newtonian behaviour of blood and can therefore not be correctly assessed in these simulations. The transition from Newtonian to non-Newtonian behaviour occurs at $\text{WSS} \leq 0.35 \text{ Pa}$ using $\mu = 3.5 \times 10^{-3} \text{ Pas}$. High WSS threshold occurs at shear rates $\sim 5000 \text{ s}^{-1}$ [56] which correlates to WSS of 17.5 Pa. Time-averaged WSS and oscillatory shear index [57] are better indicators than WSS, however, are far too difficult to calculate for a dynamic case and hence voided.

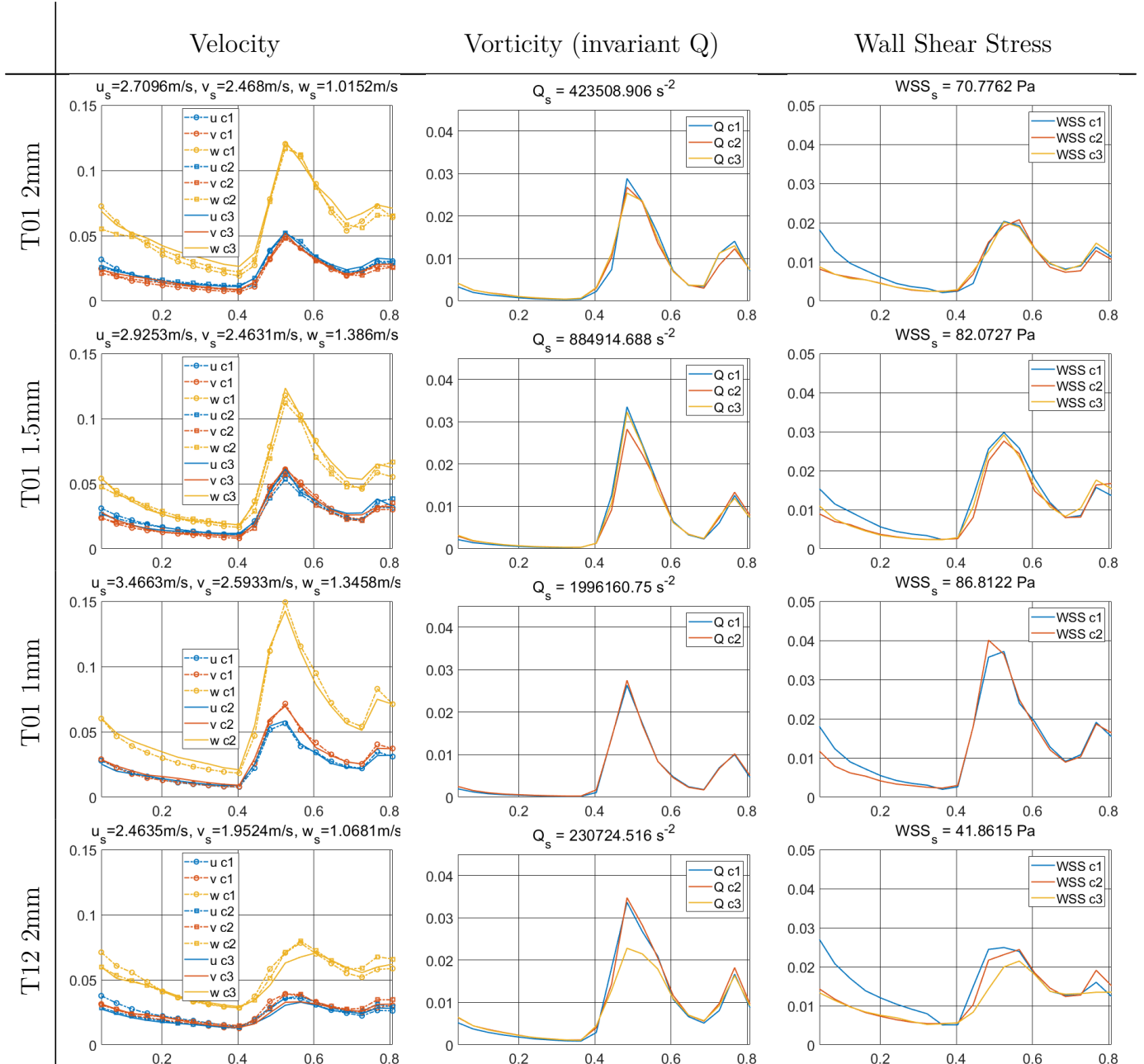
$$\tau_w = \mu(\dot{\gamma})\dot{\gamma}|_w \quad (5.3)$$

5.3: Wall shear stress for fluids [58]. τ_w is the WSS, μ the dynamic viscosity and $\dot{\gamma}$ the shear rate. μ is a function of $\dot{\gamma}$ for non-Newtonian fluids and constant for Newtonian.

5.1 Model Verification

Before results are presented, it is important to evaluate the quality of the simulation model and identify its weaknesses which may be sources of error. Similar to test cases in chapter 4.8.2, the convergence of parameters in the simulations are evaluated by RMSE

between each cycle and, for static cases, mesh convergence. The static cases with the lowest RMSE will be used for further evaluation of results. Scaled RMSE is calculated for velocity components, vorticity and WSS, which defines the flow characteristics to be evaluated. The resulting RMSE can be viewed in figure 5.1.



Continued on next page

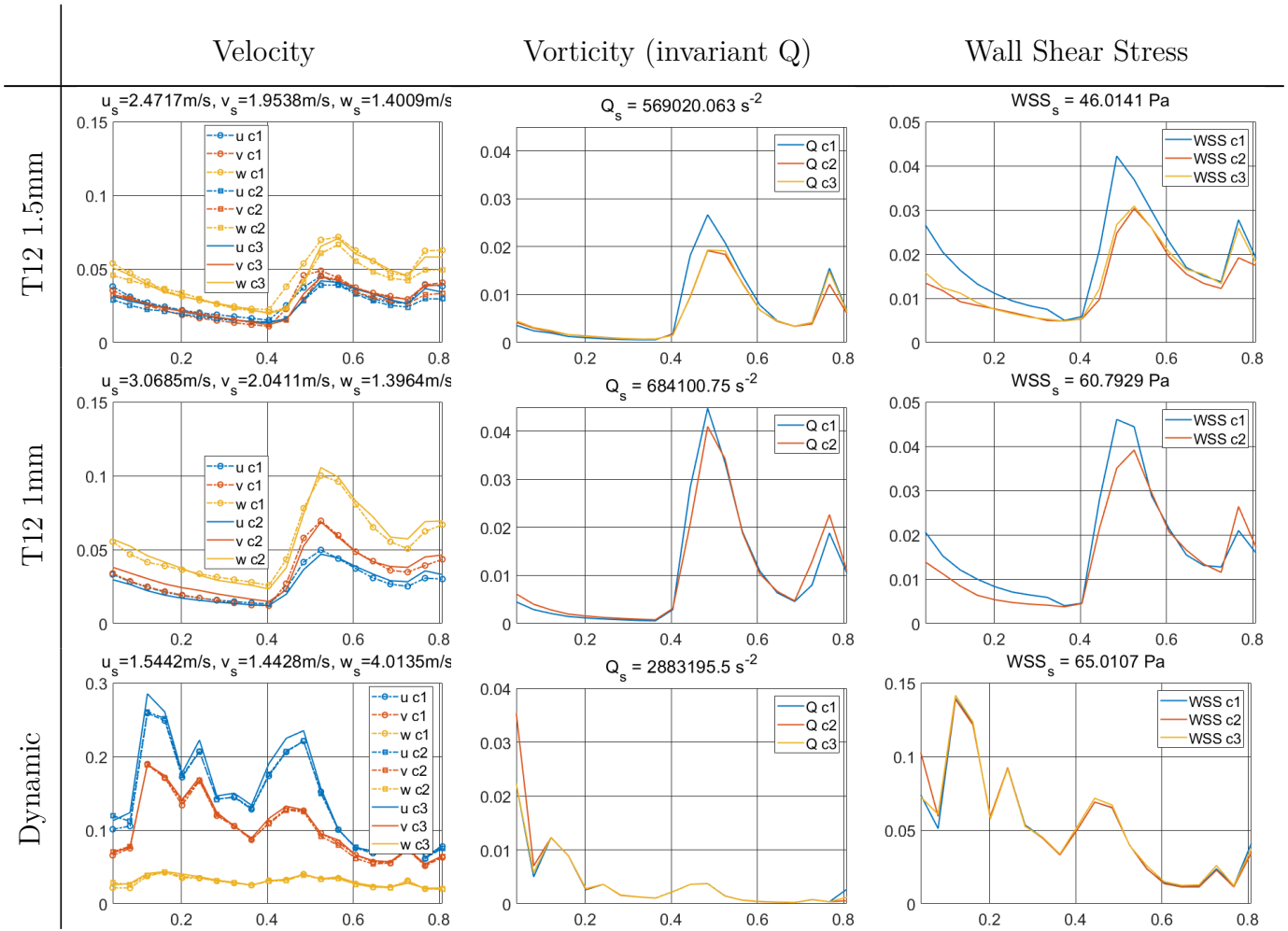


Figure 5.1: Scaled RMSE of velocity components, vorticity and WSS for all cases. RMSE is compared against the final cycle (3rd for 1 mm mesh cases and 4th for the rest) and scaled against its max value. Y-axis is the scaled RMSE with the scaling value at the top of figures with subscript s . The X-axis is the cycle time. The cycle number abbreviated as e.g. c1 = cycle 1.

Similar to the resulting RMSE of the test cases in chapter 4.8.2, the RMSE is quite high and does not converge between the cycles. The scaled RMSE does not vary in a large degree between the cases, but the scaling value does. Interestingly the scaling value increases with finer mesh for the static cases, implying a larger RMSE where the scaled RMSE is the same between cases, and thus have larger convergence difficulties than coarser mesh. Kinetic energy was also checked between the cycles which had a constant error around 10^{-4} and 10^{-3} ($\sim 10\%$ scaled error) compared to the final cycle from the 2nd cycle and out for dynamic and static cases respectively. As such, the chosen static cases for further evaluation are the 2 mm mesh scale cases. The cycle to be further evaluated

was chosen to be the final cycle, however, the second cycle would be just as good.

To further evaluate the quality of the simulation model, CFL, inlet pressure (integrated over the inlet surface) and mass conservation will be assessed for the final cycle for the selected cases. Results are shown in figure 5.2. In comparison to the RMSE result in figure 5.1 with data frequency of 20 per cycle, these results contain data from every time step in the simulation.

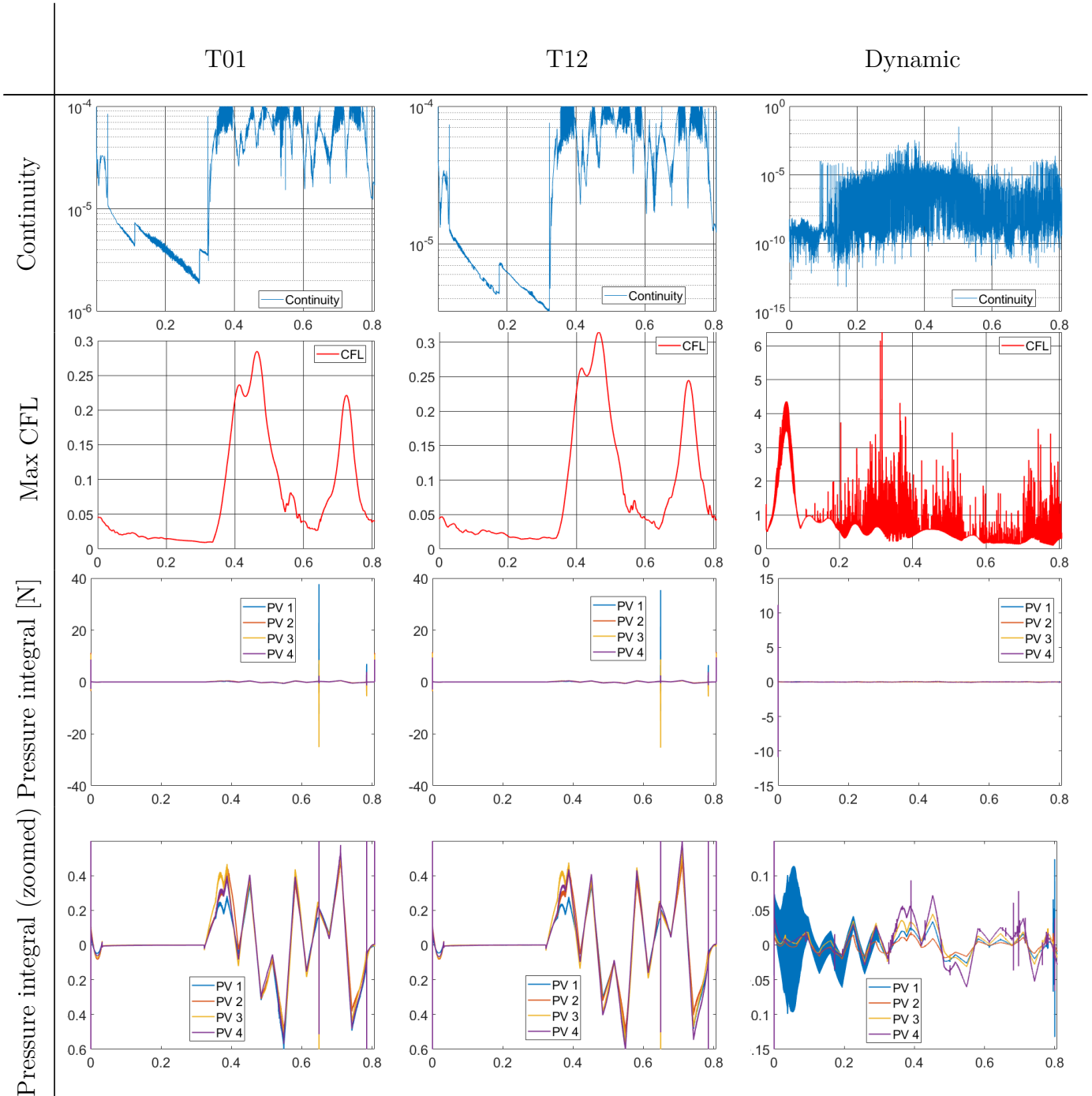


Figure 5.2: Mass conservation (continuity), CFL and inlet pressure of final cases. Y-axis are values of variable in left column. X-axis is the cycle time.

It is evident that the continuity of the static cases reaches the residual tolerance set in the simulation. The momentum residual (not shown) also met the tolerance. For the dynamic case, the residual output file became corrupted, hence the continuity was calculated from inflow, outflow and volume derivative. As such, the continuity of the dynamic case might be prone to round-off error and other inaccuracies in the calculation. It is therefore assumed that it meets the residual tolerance of 10^{-5} . Both static cases are far below the CFL limit. The dynamic case has a multitude of spikes due to remeshing. The lower outline, which would be the CFL curve if there were no spikes, is quite visible which for the most part stays below the CFL limit.

The pressure, on the other hand, shows great discontinuity. The massive spikes in row three of figure 5.2 only last for two time steps, so it is uncertain how much they impact the simulation. The two latter spikes in the static cases were found to be where the volume change compensation of the inlet flow condition transitioned from equation (4.5a) to be evenly distributed (figure 4.20(a)). This change had minuscule discontinuity, so it is strange to see such a large spike in the pressure. The first spike, also present in the dynamic case, occurs at the 2nd and 3rd time step of every cycle. The inlet flow curve is forced continuous at this region, so the cause of the spikes is unknown. In the bottom row of figure 5.2 the spikes are ignored. Interestingly the pressure integral shows linear tendency over certain intervals. For the dynamic case, there are high oscillations for one of the PVs, which probably is caused by the dynamic mesh as the pressure is integrated over a dynamic zone. All similar cases and cycles (except the 1st) exhibit the same characteristics.

During the simulation cycle, the dynamic mesh deteriorates. As shown in figure 5.3, the poor mesh quality is mainly focused around the MA outlet. This was also pointed out in the model methodology that planar adjustment caused highly skewed surfaces at the outlet. The peak poor element quality in the mesh occurs at $t = 0.36315$ s, the closest data point of the maximum skewed surface mesh at $t = 0.3551$ s and are directly linked. Improved methods of adjusting the planar outlet for the dynamic mesh should therefore be investigated.

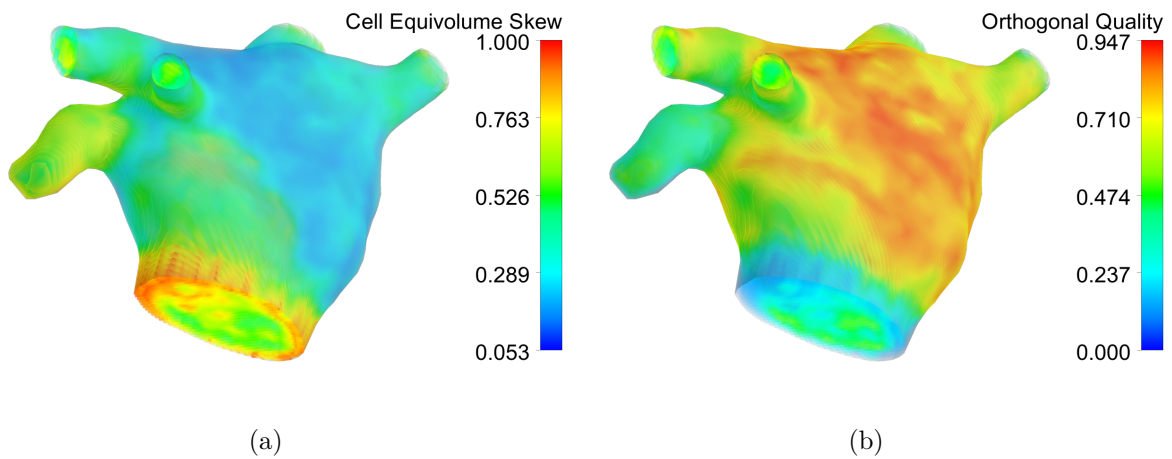


Figure 5.3: Dynamic mesh quality at $t = 0.36315$ s. Cell equivolume skew in (a) and cell orthogonal quality in (b).

5.2 Streamlines

By analysing streamlines, the flow directions are determined, and one can identify how the blood is distributed throughout the domain from a given source. In order to not overwhelm the reader with data, a select few data points have been chosen to be presented, which represents key atrial functions. These data points are illustrated in figure 5.4.

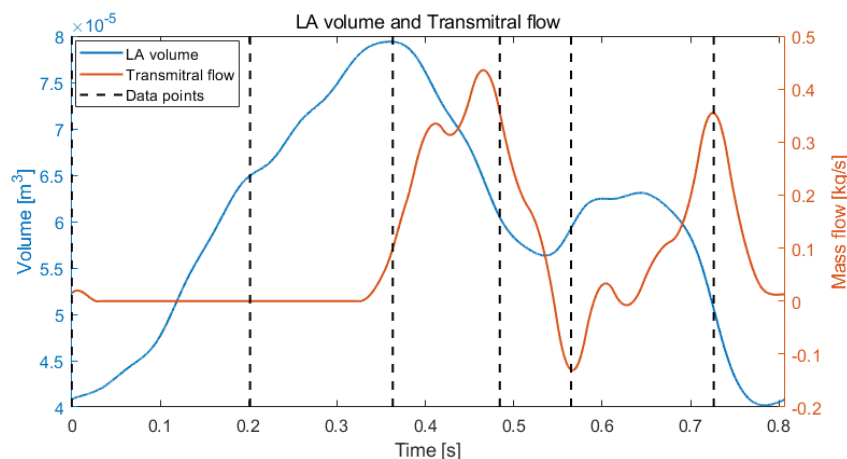


Figure 5.4: Evaluating data points in the simulation.

$t = 0.0000$ s: Transition of LV diastole to systole.

$t = 0.2018$ s: Mid LV systole.

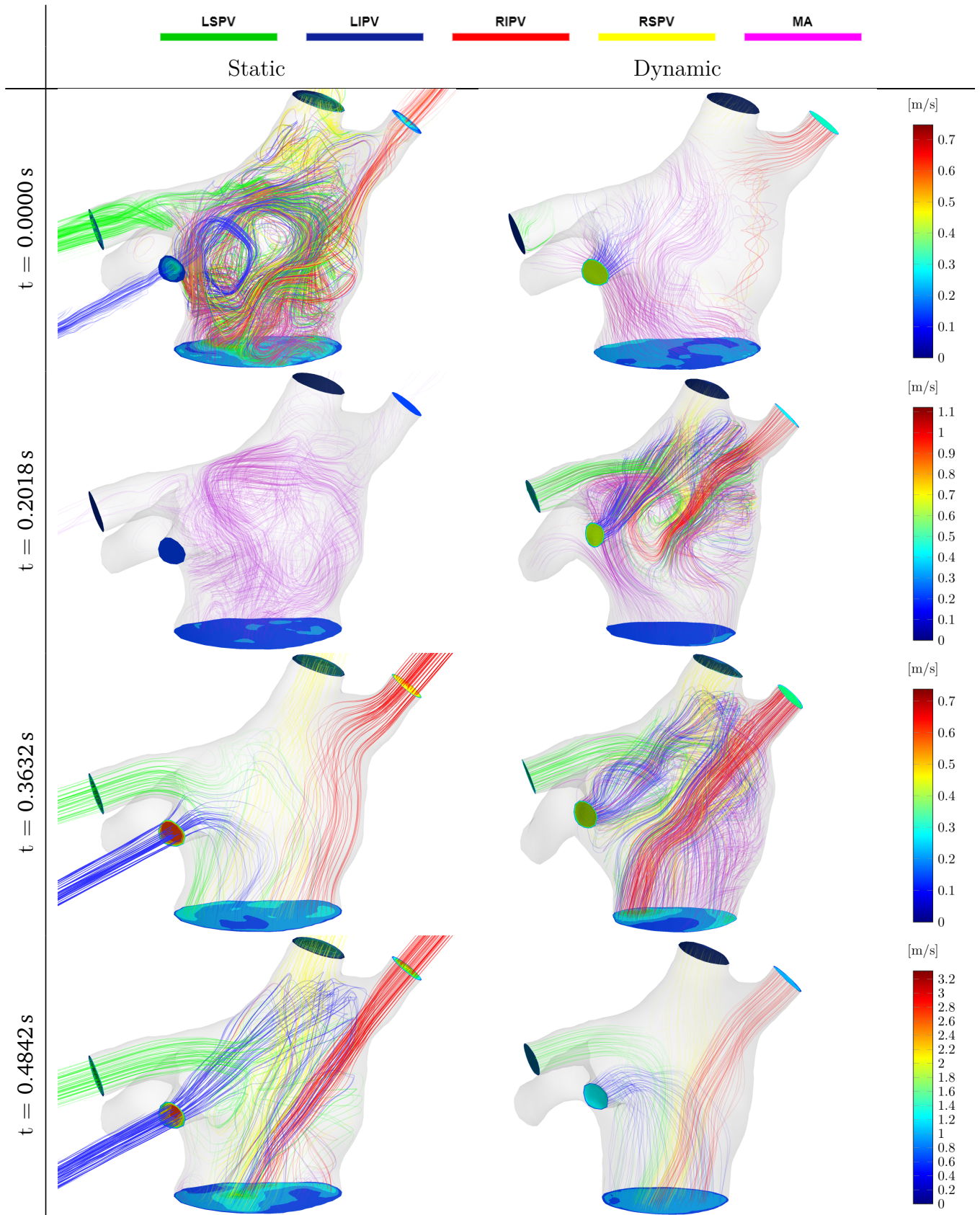
$t = 0.3632$ s: Beginning of LV diastole.

$t = 0.4842$ s: Peak (right after) E-wave transmitral flow.

$t = 0.5649$ s: Peak reverse transmitral flow.

$t = 0.7263$ s: Peak A-wave transmitral flow.

Streamlines of these data points are presented in figure 5.3, with their seeding originating from the inlet and outlet surfaces. The two static cases had very similar streamlines, hence only one of them, T01, is showed as a comparison to the dynamic case. The streamlines are colour-coded by which in- or outlet it originates from, as shown in the legend on top of the figure. The velocity profile of these surfaces is also shown in standard colour-map. Velocity magnitude of the streamlines is visualised by transparency where full opaque is the maximum velocity, and fully transparent is zero velocity. All velocities and streamlines in a row are scaled by the colour-bar to the right, where its colour-map equals the velocity profiles of the in- and outlets. While it is not the most detailed visualisation of the streamlines, it is size effective.



Continued on next page

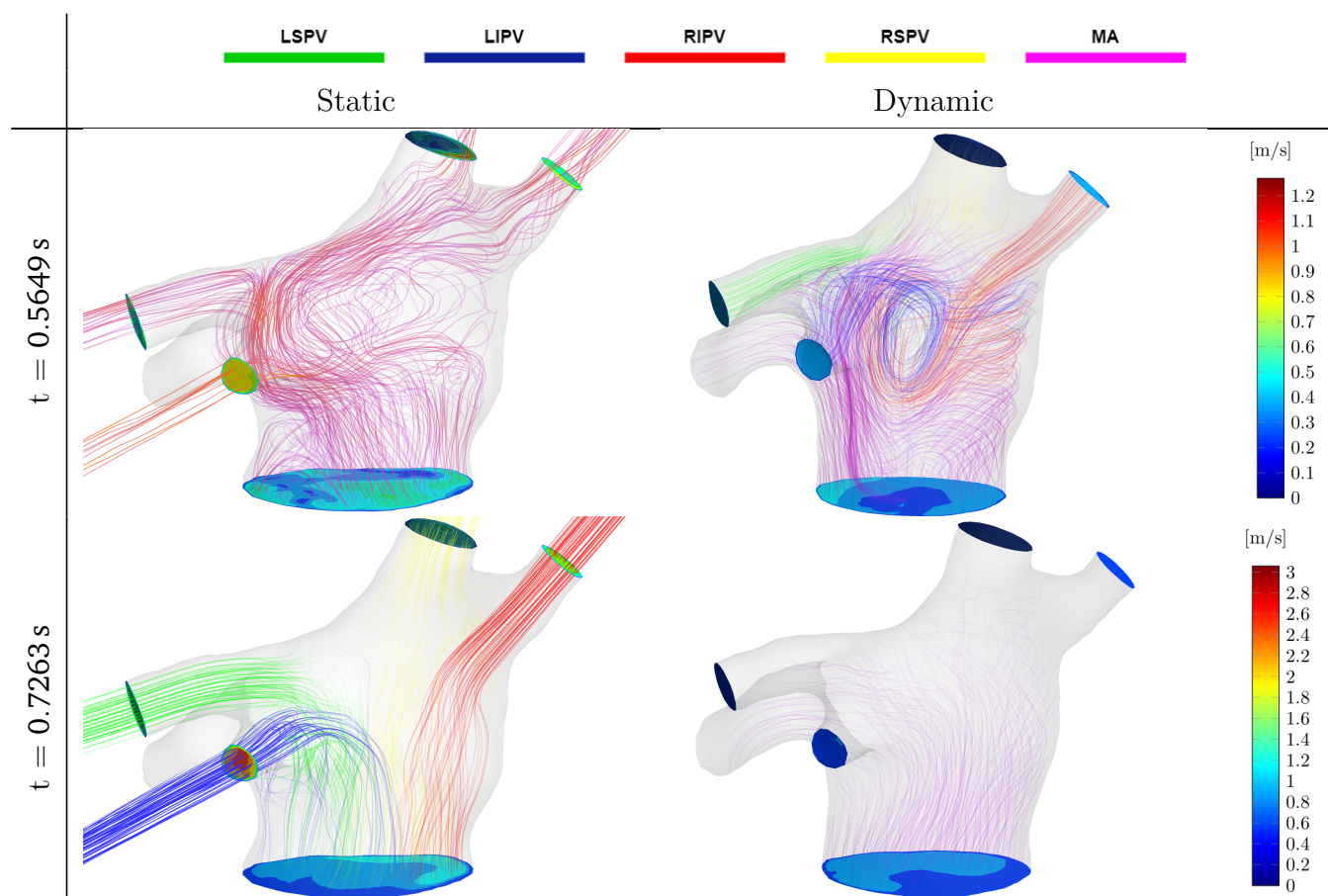


Figure 5.5: Streamlines of dynamic and T01 static cases originating from in- and outlet surfaces for six time steps. Streamlines are colour-coded by which surface it originates from, with their legends at top of the figure, and velocity magnitude visualised by transparency. Velocity profiles of in- and outlet surfaces are also visualised, with the corresponding colour-bar to the right of each row. Streamlines are also scaled by this colour-bar.

As shown from the velocity RMSE in figure 5.1, there are large variation of the velocity between the cycles, and hence variation in streamlines. As such, minuscule details of the streamlines can not be correctly assessed, but the general tendencies can. Characteristics for the time steps are:

t = 0.0000 s: The static case has high chaotic intra-atrial flow with mixing of entering flow from PVs and flow entering and leaving the MA. The dynamic case has higher PV inflow but quickly dissipates due to the expanding LA. Dynamic intra-atrial flow mainly consists of the expanding LA.

t = 0.2018 s: The static intra-atrial flow consists of residual momentum from the begin-

ning of systole. The dynamic intra-atrial flow consists of a mix of PV inflow and expanding LA, where the PV inflow fills up the upper part of the LA.

t = 0.3632 s: The static transmitral flow consists of PV inflow directed towards the MA. The dynamic transmitral flow consists of LA contraction and PV inflow, where RSPV inflow is directed straight to the MA while the other PVs are mixing in the intra-atrial flow.

t = 0.4842 s: Transmitral flow for static case consists of PV inflow and additional LA contraction for dynamic. PV inflow for static is far higher and more mixing than dynamic.

t = 0.5649 s: Reverse transmitral flow where MA inflow flows through the PVs for the static case. The dynamic case has PV and MA inflow filling up the LA expansion.

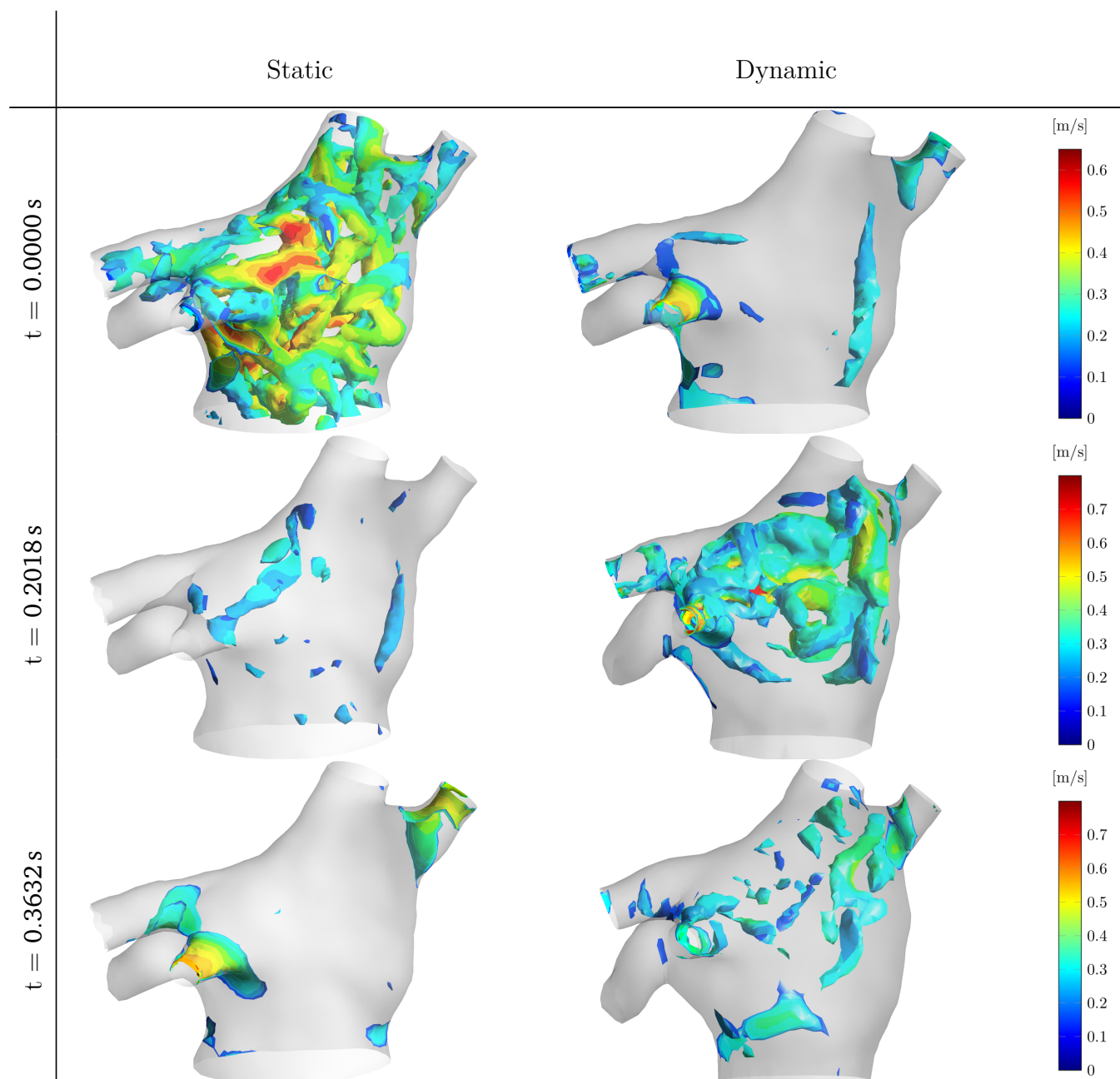
t = 0.7263 s: Transmitral flow consisting of PV inflow for the static case, and LA contraction for dynamic.

In summary, the static and dynamic cases have very different flow streamlines where the LA contraction and expansion helps lower the peak velocities from the PV inflow. The dynamic case has more stable streamlines throughout the cardiac cycle, and the static case has oscillating streamlines of high and low velocities. The static case has for some time intervals higher velocity profiles at the MA than the dynamic, implying it contains both in- and outflow. Another result which has not been discussed is that streamlines in the LAA are visible for the dynamic case and not static, implying that the LAA contraction and expansion results with higher flow inside the LAA. The dynamic case shows more agreement of streamlines with measured MRI 4D-flow fields from studies such as Suwa et al. [59].

5.3 Vortex Structures

By analysing vorticity, the smoothness of the flow field can be evaluated. High vorticity implies high swirling motion and complex flow field. Where there are mixing of streamlines in figure 5.3 there are also expected to be high vorticity. Similar to the streamline

comparison, the same data points will be evaluated and the same static case, as the static cases showed very similar tendencies. The vorticity, evaluated by the Q-criterion, is shown in figure 5.4. Structures where $Q = 1500 \text{ s}^{-2}$ are shown with velocity contours. These structures envelop a vortex core where the vorticity increases closer to the core and decreases further away from the core. Separation of flow is implied where the vortex structure envelops the wall.



Continued on next page

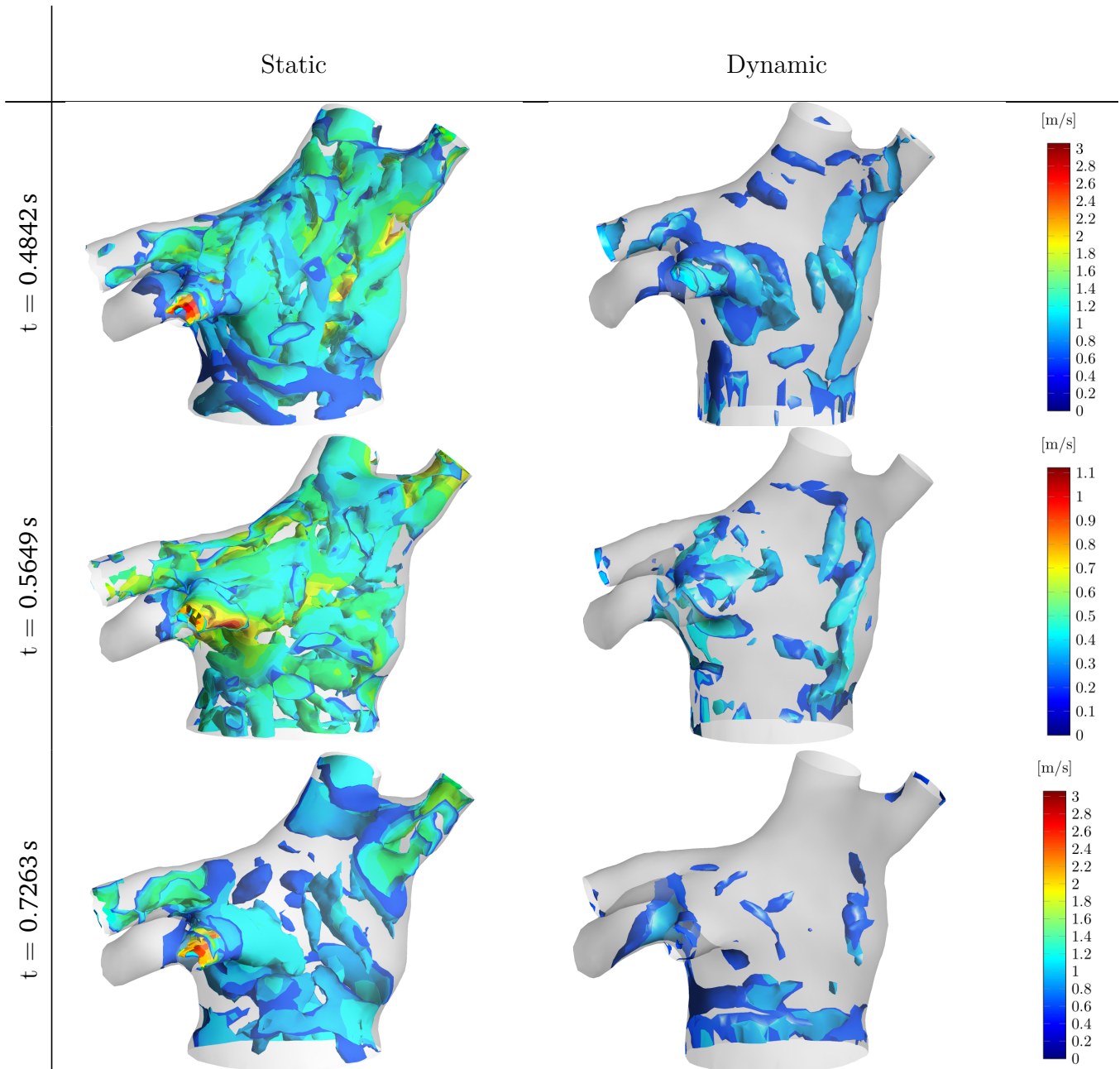


Figure 5.6: Vortex structures of $Q\text{-criterion} = 1500 \text{ s}^{-2}$ with velocity contour for six time steps.

Similar to the streamlines, the vorticity has high variance between the cycles and hence overall behaviour of vorticity is assessed. The vortex structures are coherent with the streamlines. Larger vortex structures occur where the streamlines are mixing. It is evident that the static case has a higher density of vortex structures than dynamic, indicating a more complex intra-atrial flow field. The density of dynamic vortex structures is evenly distributed throughout the cardiac cycle, usually in the upper part of the LA, while the

high density of static vortex structures occurs at high transmitral flow throughout the entire LA domain. Separation of flow usually occurs at the LIPV (PV2) inlet, and more often for static than dynamic case. It is also visible that the vortex structures occurs in the LAA for both cases.

5.4 Wall Shear Stress

With WSS there are two phenomenons being checked; $WSS \geq 17.5 \text{ Pa}$ which can cause tissue rupture and thrombus aggregation, and $WSS \leq 0.35 \text{ Pa}$ where the Newtonian model is no longer valid. As the WSS is proportional to the strain rate, high velocity near the wall leads to high WSS and vice versa. The WSS is presented as a fraction of area impacted by the WSS criteria to the total surface area of the LA. High and low WSS are shown in figure 5.8. Both static cases and the dynamic are shown with all data sets from the simulation. The extensions of the static LA domain are not included in the calculation.

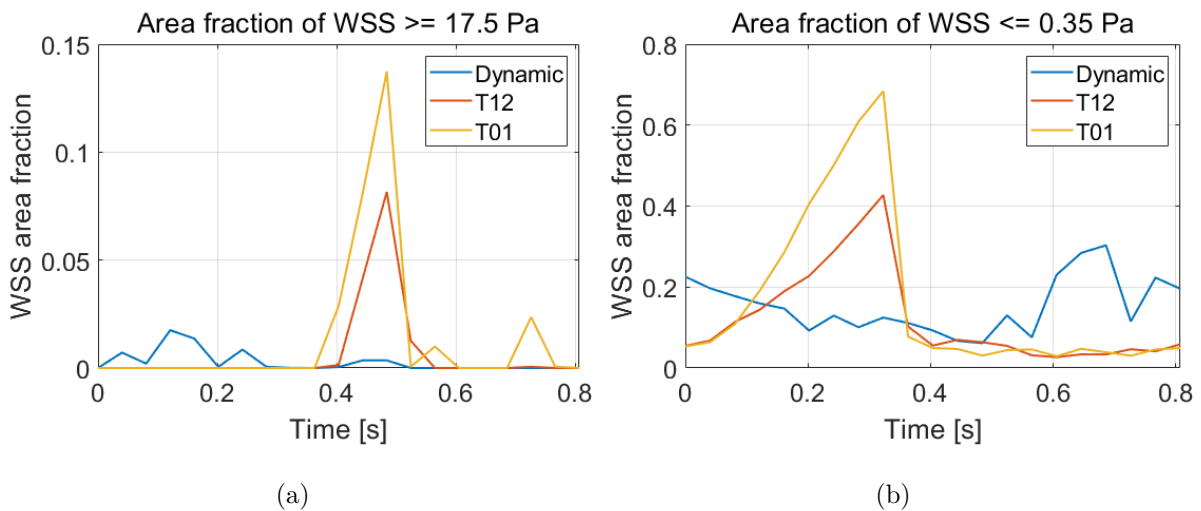


Figure 5.7: Area fraction of high and low WSS criteria to the total LA surface area. High WSS ($\geq 17.5 \text{ Pa}$) in (a) and low WSS ($\leq 0.35 \text{ Pa}$) in (b).

As shown, the static cases have high spikes of high and low WSS, the dynamic has more evenly distributed WSS, and the T01 static case has higher peaks than T12. The spike for $WSS \geq 17.5 \text{ Pa}$ occurs at the E-wave of the transmitral flow, which is the highest peak. The spike for $WSS \leq 0.35 \text{ Pa}$ is unsurprisingly due to no inlet or outlet flow for

the static cases during LV systole, and rapidly decreases once the diastole begins. The low WSS curve of the static cases also corresponds well with the vorticity. If there is high vorticity, there will also be less low WSS area.

Further investigation of the WSS reveals that the high WSS occurs at the PV inlets, usually LIPV, and the peak of the static cases occurs at the backside of the LA due to high LIPV inflow, as shown in figure 5.8(a). The peak of low WSS for the static cases are shown in figure 5.8(b) where the WSS is evenly distributed.

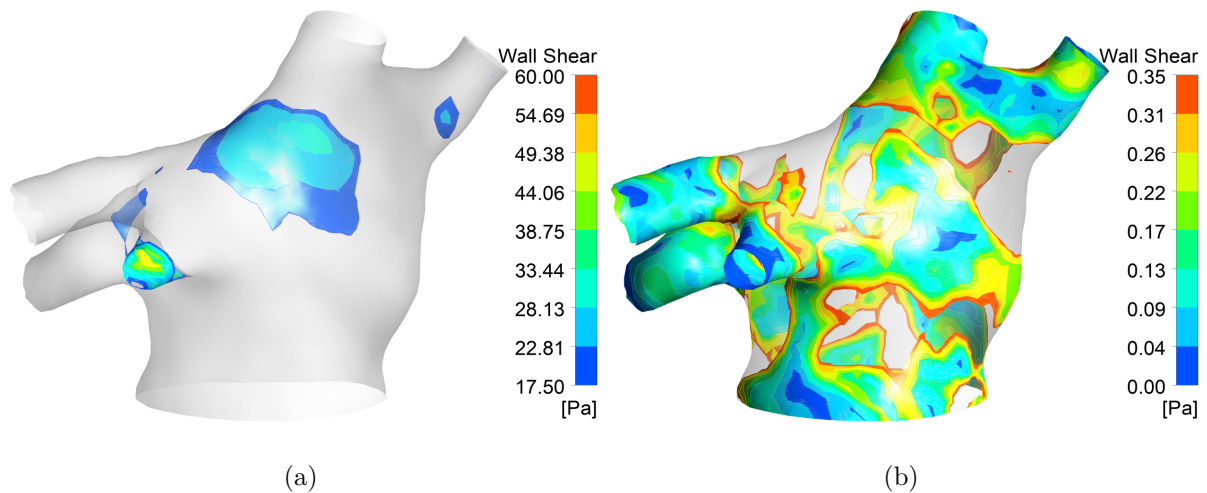


Figure 5.8: Peak high static WSS area (a) at $t = 0.44385$ s and peak low static WSS area (b) at $t = 0.3228$ s. Shown static case is T01.

More interestingly is the evaluation of low WSS in the LAA for static and dynamic cases. For dynamic cases, the low WSS area is continuously shifting throughout the cycle, even in the LAA due to the dynamic domain. For the static cases, there is a constant area of low WSS in the tip of the LAA, implying that there may be residence of blood over multiple cardiac cycles and hence risk of thrombus to occur. As such, the static case may over predict the thrombus formation in thrombus models, such as Wang et al. [33]. Figure 5.5 shows the low WSS for dynamic and static cases for two time steps; beginning of systole and mid diastole which represents how the low WSS begins in the cardiac cycle, and conveniently where the static case has lowest WSS area and the dynamic highest WSS area.

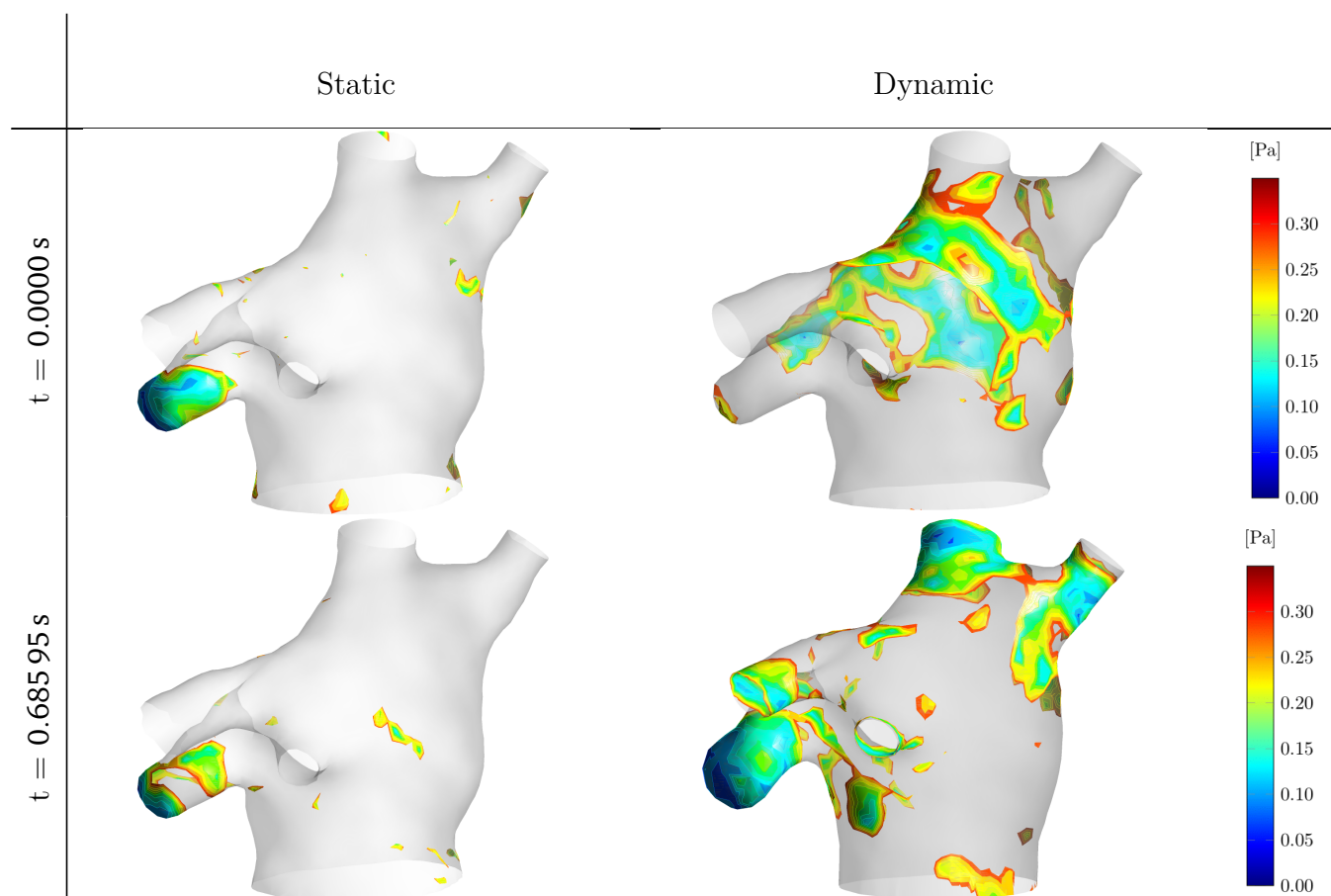


Figure 5.9: Change in low WSS area of static and dynamic cases. Static cases have a constant low WSS area in the LAA while the dynamic is varying over the entire LA surface. Shown static case is T01.

As a final check of the results, the same presented characteristics were analysed for the previous cycle. The characteristics had the same tendencies of streamline and vortex structure behaviour and WSS density, however, with slightly different values and positions. As such, the comparison of characteristics between the cases is the same between the cycles.

5.5 Results Discussion

As shown, the static and dynamic cases exhibit very different behaviour for the intra-atrial flow. While the dynamic case has a quite stable density of WSS and vortex structures, the static case have large variations. This can also be seen in the streamlines where the static case has a more complex flow field than the dynamic. For the two static cases,

the characteristics were very similar. The only major difference is the peak of high and low WSS area between the cases. However, this is believed to be caused by the slightly different PV inlet geometry, notably LIPV, which has a slight difference of cross-sectional area and causes higher Re, shown in figure 4.21 in chapter 4.6.3. This variance between the static cases is believed to be within the area of RMSE, i.e. that with the numerical error in the simulation, the difference can not be correctly assessed.

The results, however, vary between the cycles as shown in the RMSE. As such, a stable cyclic solution can not be correctly assessed in detail, but the general behaviour of the different cases can. An interesting result of the mesh convergence study performed on the static cases is that the finer mesh showed higher tendencies for variance between the cycles. The different mesh cases were run with the same time step size, however with $CFL \leq 1$. This may indicate that the solution is dependent on even lower CFL to reach cyclic convergence, as the coarser mesh had lower CFL and better convergence.

For both cases, the area of strain rate for non-Newtonian behaviour of blood is substantial. In order to more correctly assess the intra-atrial flow, with special regard to thrombus formation, non-Newtonian models of blood should be investigated. There are numerous hemodynamical non-Newtonian models, as demonstrated by Doost et al. [38], which must be considered with care. The static model has higher turbulent inflow than the dynamic model. While the dynamic inflow barely reaches turbulent behaviour, and the laminar model can be assumed, turbulence models should be evaluated for the static model. This is again very dependent on correct flow measurements as the turbulent flow is highly based on these measurements.

With the high cyclic RMSE, inlet pressure inconsistency, high dynamic CFL and areas of non-Newtonian behaviour, the results are not expected to mimic the details of the actual intra-atrial flow correctly. Without measurements to validate the results, it is difficult to assess how the results correlate to actual intra-atrial flow, but the differences in static and dynamic results are believed to be reasonable. The dynamic flow field shows more agreement with other studies, however, the comparison is superficial.

5.6 Simulation Model Discussion

The most significant error in the simulation model is, of course, the fact that the transient PV flow data originates from a different patient. As the model is patient-specific, and LA geometry, especially the LAA, can vary in a large degree between patients, the PV flow is also expected to vary. The systolic flow, however, is expected to be quite accurate as it is calculated by the volume change in the LA. The distribution of flow from the PVs are only affected by the flow data during systole. It is uncertain how the actual flow data, should it be used for the same simulations in the future, impacts the behaviour of the resulting characteristics, but it is believed that similar differences of static and dynamic cases will prevail.

5.6.1 Boundary Conditions

The integral of pressure at the PV inlets shows massive spikes at certain intervals. The most believable explanation is that the inlet flow curves are non-continuous at these intervals. Unfortunately, there was no time to investigate the exact cause further. For the outlet, it is unknown how the resulting transmitral flow profile resembles the actual flow profile. Based on flow measurements at the MA, a more detailed description of the transmitral flow profile at the MA could be introduced. This could be implemented as a pressure profile if the same inlet boundary conditions are kept. If both outlet and inlet boundary conditions are determined by mass flow, the model will be over-constrained and experience issues with mass conservation.

5.6.2 Segmentation

It is uncertain how accurate the segmented LA geometry represents the anatomically correct geometry. The largest uncertainties are the PVs and LAA which have smaller dimensions than the LA and hence more difficult to segment with the current MRI resolution. Most notably is the LIPV (PV2) which in this thesis is segmented with a lower cross-sectional area than commonly observed. Acquiring MRI axes which are perpendicular to each other could improve the resolution quality of the combined volume. As the PVs are highly patient-specific, the segmentation could, of course, be correct. The MRI

spatial resolution, slice thickness and slice alignment can be improved by CT, which has a higher spatial resolution, however, with lower temporal resolution. Segmentation of CT in other studies shows far higher detailed geometry, and it is therefore believed that CT will improve the geometrical accuracy. Whether the increased spatial resolution weighs up for the temporal resolution must therefore be considered. Segmentation from MRI is also extremely time-consuming, especially when creating a dynamic model, where it is mostly drawn by hand, while with CT, it can be close to fully automated. It must be noted that CT does not have the possibility of flow measurements, hence flow measurements of the PVs must be performed with MRI either way.

5.6.3 MV Modelling

The MV has been entirely neglected in the model, and a zero reference uniform pressure profile has been used at the MA outlet. It is uncertain how a more detailed description of the MV boundary condition would affect the simulation results. However, it is believed that it would primarily influence the result of the lower part of the LA and not necessarily the PVs and LAA. It is uncertain how a more detailed MV modelling could be introduced to a LA model separated from the LV. By introducing an FSI model, flow below the MV must be included, i.e. that the LV must be included.

5.6.4 Dynamic Mesh

The most problematic part of the dynamic mesh in this thesis is the planar outlet and inlets, which caused poor quality mesh and geometrical errors that had to be corrected. It also limited how small the time step and mesh size in the simulation could be set, even without enforcing planar surfaces at the inlets. This issue could result from the displacement field generated by image registration in Slicer which other softwares may not exhibit. Other methods could potentially be used to generate the dynamic surface mesh but are currently unknown to the author.

With the current method to generate the dynamic surface mesh by controlling the surface nodes, extensions of the outlet and inlets could not be introduced, which allows the velocity profile to develop for the inlets. With the extensions, the inlet and outlets could be placed outside the dynamic zone and planar enforcement could be voided, hence

improving the dynamic mesh stability. This could also improve the inlet pressure consistency of the dynamic simulation. This issue could potentially be entirely voided if the displacement field is directly implemented into ANSYS Fluent. The surface mesh could then be able to remesh, hence improving the mesh quality. The outlet extension of the MA have higher displacements than the inlets and thereby introduce more difficulties with the dynamic mesh than the inlet extensions. How this issue should be approached is unknown to the author.

The dynamic mesh is based on image registration where the displacement field is determined by best fit of difference between images. This does not necessarily mimic the actual displacement of the LA wall. MRI tagging and Speckle tracking echocardiography could improve mesh displacement.

Chapter 6

Conclusion and Further Work

By comparing the CFD simulations of the static and dynamic LA models reveals substantial differences in the streamlines, vortex structures and WSS of the intra-atrial flow. The static model shows a highly varying chaotic and complex flow field. In contrast, the dynamic model is quite smooth and consistent with a higher agreement with measured intra-atrial flow field in other studies. There are also indications that the static model will over predict thrombus formation in the LAA, while the expansion and contraction of the LAA in the dynamic model prevents the blood stasis. Two different static models were also run, where the geometry represented the minimum and maximum LA volume in the cardiac cycle. Results show that these cases were practically identical in their characteristics. For further evaluation of LA CFD analysis, should it be viable to predict LA CVDs in clinical practice, it is therefore highly recommended to use a dynamic model. If a static model is used for initial proof of concepts, the chosen time step of the LA geometry is arbitrary.

Based on the work in this thesis, further investigation of cyclic stability in the flow field must be performed in order for LA CFD results to be realistic. This primarily consists of time step and mesh convergence. Also, substantial areas of non-Newtonian behaviour are found in the simulation, hence a non-Newtonian model must be used in order to correctly assess the intra-atrial flow field and residence time of blood. The static case also exhibits significant turbulent behaviour, and a low-Reynolds number turbulence model such as the RANS $k-\omega$ SST should therefore be considered. This is, however, based on flow measurements not corresponding with the patient-specific model.

For the simulation model presented in this thesis, the following suggestions are recommended to improve the model:

- Acquire correct flow measurements for inlet boundary conditions.
- Acquire MA flow measurements to impose a pressure profile for the MA outlet boundary condition.
- Acquire intra-atrial flow measurements to validate the results.
- Investigate if non-continuous flow curves cause inlet pressure inconsistency.
- Perform segmentation based on CT to improve the geometrical accuracy and possibly automate the segmentation process. Must be weighed up against the loss of temporal resolution.
- Find better techniques to generate the dynamic surface mesh. This includes:
 - Investigate other software to perform image registration or other methods to generate the displacement field.
 - Investigate if MRI tagging or Speckle tracking echocardiography can be used for generating the displacement field to acquire a more accurate mesh deformation.
 - Other methods to enforce planar inlets and outlet to improve the surface mesh quality.
 - Generate displacement field which is compatible with boundary extensions or new methods to include the extensions.
 - Implement displacement fields directly into the CFD software to allow the dynamic surface mesh to remesh.

Bibliography

- [1] E. Wilkins, L. Wilson, K. Wickramasinghe, P. Bhatnagar, J. Leal, R. Luengo-Fernandez, R. Burns, M. Rayner and N. Townsend, *European Cardiovascular Disease Statistics 2017*. European Heart Network, 2017, pp. 8–9.
- [2] P. K. et al., ‘2016 esc guidelines for the management of atrial fibrillation developed in collaboration with eacts’, *European Heart Journal*, vol. 37, no. 38, pp. 2893–2962, 2016. doi: 10.1093/eurheartj/ehw210.
- [3] R. A. Gray and P. Pathmanathan, ‘Patient-specific cardiovascular computational modeling: Diversity of personalization and challenges’, *Journal of Cardiovascular Translational Research*, vol. 11, no. 2, pp. 80–88, 2018. doi: 10.1007/s12265-018-9792-2.
- [4] K. Moore and E. Roth, *Heart failure*, <https://www.healthline.com/health/heart-failure>, (accessed: 29.06.2020).
- [5] V. S. Skjefstad, ‘Inner surface modeling of the left ventricle and atrium: From MRI images to moving mesh’, *Unpublished*, 2019.
- [6] E. Lifesciences, *Basic anatomy and function of the heart*, <https://yourheartvalve.com/heart-basics/heart-anatomy/>, (accessed: 29.06.2020).
- [7] T. O. University, *Non-communicable diseases, emergency care and mental health*, https://www.open.edu/openlearncreate/mod/oucontent/view.php?id=287&extra=thumbnail_idp2934720, (accessed: 29.06.2020).
- [8] W. Hamilton, *Textbook of the Human Anatomy*. The MacMillan Press LTD, 1976, isbn: 978-0-333-34029-5. doi: 10.1007/978-1-349-06486-1.
- [9] F. M. Filipoiu, *Atlas of Heart Anatomy and Development*. Springer, 2014, isbn: 978-1-4471-5381-8. doi: 10.1007/978-1-4471-5382-5.
- [10] J. E. Hall, *Guyton and Hall Textbook of Medical Physiology - 12th-Ed*. Saunders, 2019.
- [11] R. E. Klabunde, *Cardiovascular Physiology Concepts*. Lippincott Williams Wilkins, 2012, isbn: 9781451113846.
- [12] R. Beigel, N. C. Wunderlich, S. Y. Ho, R. Arsanjani and R. J. Siegel, ‘The left atrial appendage: Anatomy, function, and noninvasive evaluation’, *JACC: Cardiovascular Imaging*, vol. 7, no. 12, pp. 1251–1265, 2014. doi: 10.1016/j.jcmg.2014.08.009.
- [13] A. Kandathil and M. Chamarthi, ‘Pulmonary vascular anatomy anatomical variants’, *Cardiovasc Diagn Ther*, vol. 8, no. 3, pp. 201–207, 2018.
- [14] R. B. Saeed, *Heart chambers*, <https://cardiovascularsystemud.weebly.com/the-heart1.html>, (accessed: 29.06.2020).
- [15] M. V. R. Center, *Mitral annulus*, <https://www.mitralvalverepair.org/mitral-annulus>, (accessed: 29.06.2020).

- [16] S. Y. Ho, 'Anatomy of the mitral valve', *Heart*, vol. 88, no. 4, pp. 5–10, 2002.
- [17] J. R. Mitchell and J.-J. Wang, 'Expanding application of the wiggers diagram to teach cardiovascular physiology', *Adv Physiol Educ*, vol. 38, no. 2, pp. 170–175, 2014. doi: 10.1152/advan.00123.2013.
- [18] Wikipedia, *Wiggers diagram*, https://en.wikipedia.org/wiki/Wiggers_diagram, (accessed: 29.06.2020).
- [19] K. Linden, F. Goldschmidt, K. T. Laser, C. Winkler, H. Körperich, R. Dalla-Pozza, J. Breuer and U. Herberg, 'Left atrial volumes and phasic function in healthy children: Reference values using real-time three-dimensional echocardiography', *Journal of the American Society of Echocardiography*, vol. 32, no. 8, pp. 1036–1045, 2019. doi: 10.1016/j.echo.2019.03.018.
- [20] O. K. Baskurt and H. J. Meiselman, 'Blood rheology and hemodynamics', *Semin Thromb Hemost*, vol. 29, no. 5, pp. 435–450, 2003. doi: 10.1055/s-2003-44551.
- [21] NHLBI, *Cardiomyopathy*, <https://www.nlm.nih.gov/health-topics/cardiomyopathy>, (accessed: 29.06.2020).
- [22] M. Clinic, *Cardiomyopathy*, <https://www.mayoclinic.org/diseases-conditions/cardiomyopathy/symptoms-causes/syc-20370709>, (accessed: 29.06.2020).
- [23] NHLBI, *Arrhythmia*, <https://www.nlm.nih.gov/health-topics/arrhythmia>, (accessed: 29.06.2020).
- [24] R. Delewi, F. Zijlstra and J. J. Piek, 'Left ventricular thrombus formation after acute myocardial infarction', *Heart*, vol. 98, pp. 1743–1749, 2012. doi: 10.1136/heartjnl-2012-301962.
- [25] W.-C. L. et al, 'Left atrial or left atrial appendage thrombus resolution after adjustment of oral anticoagulant treatment', *Journal of Stroke and Cerebrovascular Diseases*, vol. 28, no. 1, pp. 90–96, 2019.
- [26] M. Tee, J. A. Noble and D. A. Bluemke, 'Imaging techniques for cardiac strain and deformation: Comparison of echocardiography, cardiac magnetic resonance and cardiac computed tomography', *Expert Review of Cardiovascular Therapy*, vol. 11, no. 2, pp. 221–231, 2013.
- [27] J. C. Morud, P. Skjetne, S. Urheim and S. K. Dahl, 'The effect of chordae tendineae on systolic flow', *Computers in Biology and Medicine*, vol. 109, pp. 91–100, 2019.
- [28] J. L. et al., 'Patient-specific simulation of cardiac blood flow from high-resolution computed tomography', *Journal of Biomechanical Engineering*, vol. 138, no. 12, 2016. doi: 10.1115/1.4034652.
- [29] B. S. et al., 'Effects of left atrium on intraventricular flow in numerical simulations', *Computers in Biology and Medicine*, vol. 106, pp. 46–53, 2019. doi: 10.1115/1.4034652.
- [30] T. Otani, A. Al-Issa, A. Pourmorteza, E. R. McVeigh, S. Wada and H. Ashikaga, 'A computational framework for personalized blood flow analysis in the human left atrium', *Annals of Biomedical Engineering*, vol. 44, no. 11, pp. 3284–3294, 2016. doi: 10.1007/s10439-016-1590-x.
- [31] A. Masci, M. Alessandrini, D. Forti, H. F. Menghini, L. Dede, C. Tomasi, A. Quarteroni and C. Corsi, 'A proof of concept for computational fluid dynamic analysis of the left atrium in atrial fibrillation on a patientspecific basis', *Journal of Biomechanical Engineering*, vol. 142, 2019. doi: 10.1115/1.4044583.

-
- [32] R. K. et al., ‘Numerical analysis of hemodynamic changes in the left atrium due to atrial fibrillation’, *Journal of Biomechanics*, vol. 48, no. 3, pp. 472–478, 2015. doi: 10.1016/j.jbiomech.2014.12.025.
- [33] Y. Wang, Y. Qiao, Y. Mao, C. Jiang, J. Fan and K. Luo, ‘Numerical prediction of thrombosis risk in left atrium under atrial fibrillation’, *Mathematical Biosciences and Engineering*, vol. 17, no. 3, pp. 2348–2360, 2020. doi: 10.3934/mbe.2020125.
- [34] S. K. Dahl, E. Thomassen, L. R. Hellevik and B. Skallerud, ‘Impact of pulmonary venous locations on the intra-atrial flow and the mitral valve plane velocity profile’, *Cardiovascular Engineering and Technology*, vol. 3, no. 3, pp. 269–281, 2012.
- [35] J. L. et al., ‘Impact of pulmonary venous inflow on cardiac flow simulations: Comparison with in vivo 4d flow mri’, *Annals of Biomedical Engineering*, vol. 47, no. 2, pp. 413–424, 2019. doi: 10.1007/s10439-018-02153-5.
- [36] M. W., C. A., M. R., P. C. and S. W., ‘Fully-coupled fluid-structure interaction simulation of the aortic and mitral valves in a realistic 3d left ventricle model’, *PLoS ONE*, vol. 12, no. 9, 2017. doi: 10.1371/journal.pone.0184729.
- [37] B. Skallerud, V. Prot and I. S. Nordrum, ‘Modeling active muscle contraction in mitral valve leaflets during systole: A first approach’, *Biomech Model Mechanobiol*, vol. 10, pp. 11–26, 2011. doi: 10.1007/s10237-010-0215-9.
- [38] S. N. Doost, L. Zhong, B. Su and Y. S. Morsi, ‘The numerical analysis of non-newtonian blood flow in human patient-specific left ventricle’, *Computer Methods and Programs in Biomedicine*, vol. 127, pp. 232–247, 2016. doi: 10.1016/j.cmpb.2015.12.020.
- [39] C. Chnafa, S. Mendez and F. Nicoud, ‘Image-based large-eddy simulation in a realistic left heart’, *Computers Fluids*, vol. 94, no. 1, pp. 173–187, 2014.
- [40] Slicer 4.10.2 r28257, <https://www.slicer.org/>.
- [41] Y.-H. Kim, E. M. Marom, J. E. H. II and H. P. McAdams, ‘Pulmonary vein diameter, cross-sectional area, and shape: Ct analysis’, *Radiology*, vol. 235, no. 1, pp. 43–50, 2005. doi: 10.1148/radiol.2351032106.
- [42] E. H. W. Meijering, W. J. Niessen, J. P. W. Pluim and M. A. Viergever, ‘Quantitative comparison of sinc-approximating kernels for medical image interpolation’, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI’99*, Springer Berlin Heidelberg, 1999, pp. 210–217, isbn: 978-3-540-48232-1.
- [43] S. Mihaila, D. Muraru, E. Piasentini, M. H. Miglioranza, D. Peluso, U. Cucchini, S. Iliceto, D. Vinereanu and L. P. Badano, ‘Quantitative analysis of mitral annular geometry and function in healthy volunteers using transthoracic three-dimensional echocardiography’, *Journal of the American Society of Echocardiography*, vol. 27, no. 8, pp. 846–857, 2014. doi: 10.1016/j.echo.2014.04.017.
- [44] MATLAB Release R2019b, The MathWorks, Inc., Natick, Massachusetts, United States.
- [45] ANSYS Fluent 19.2, *Fluent User Guide, Part II: Solution Mode, Chapter 10.6*, (accessed: 11.06.2020).
- [46] H.-B. Kim, J. Hertzberg, C. Lanning and R. Shandas, ‘Noninvasive measurement of steady and pulsating velocity profiles and shear rates in arteries using echo piv: In vitro validation studies’, *Annals of Biomedical Engineering*, vol. 32, no. 8, pp. 1067–1076, 2004. doi: 10.1114/B:ABME.0000036643.45452.6d.

- [47] M.-Y. Jeung, P. Germain, P. Croisille, S. E. Ghannudi, C. Roy and A. Gangi, ‘Myocardial tagging with mr imaging: Overview of normal and pathologic findings’, *RadioGraphics*, vol. 32, no. 1, pp. 1381–1398, 2012. doi: 10.1148/rg.325115098.
- [48] B. H. Amundsen, T. Helle-Valle, T. Edvardsen, H. Torp, J. Crosby, E. Lyseggen, A. Støylen, H. Ihlen, J. A. Lima, O. A. Smiseth and S. A. Slørdahl, ‘Correlation between laa morphological features and computational fluid dynamics analysis for non-valvular atrial fibrillation patients’, *Applied Sciences*, vol. 10, no. 4, p. 1448, 2020. doi: 10.3390/app10041448.
- [49] Y. A. Cengel and J. M. Cimbala, *Fluid Mechanics*. McGraw-Hill Education, 2014, pp. 350–353, isbn: 978-1-259-01122-1.
- [50] S. Liu and H. Tsai, ‘Simulation of boundary layer transition with a modified k-omega model’, *American Institute of Aeronautics and Astronautics*, 1998. doi: 10.2514/6.1998-340.
- [51] ANSYS Fluent 19.2, *Fluent Theory Guide, Chapter 1,3,21*, (accessed: 11.06.2020).
- [52] R. Courant, K. Friedrichs and H. Lewy, ‘On the partial difference equations of mathematical physics’, *IBM Journal of Research and Development*, vol. 11, no. 2, pp. 215–234, 1967. doi: 10.1147/rd.112.0215.
- [53] ANSYS Fluent 19.2, *Fluent User Guide, Part II: Solution Mode, Chapter 29*, (accessed: 11.06.2020).
- [54] J.-m. Zhan, Y.-t. Li, W.-h. O. Wai and W.-q. Hu, ‘Comparison between the q criterion and vortex in the application of an in-stream structure’, *Physics of Fluids*, vol. 31, no. 12, 2019. doi: 10.1063/1.5124245.
- [55] D. M. Wootton and D. N. Ku, ‘Fluid mechanics of vascular systems, diseases, and thrombosis’, *Annu. Rev. Biomed. Eng.*, vol. 1, pp. 299–329, 1999.
- [56] L. D. Casa, D. H. Deaton and D. N., ‘Role of high shear rate in thrombosis’, *Journal of Vascular Surgery*, vol. 61, no. 4, pp. 1068–1080, 2015. doi: 10.1016/j.jvs.2014.12.050.
- [57] D. Martin, E. Murphy and F. Boyle, ‘Computational fluid dynamics analysis of balloon-expandable coronary stents: Influence of stent and vessel deformation’, *Medical Engineering Physics*, 2014. doi: 10.1016/j.medengphy.2014.05.011.
- [58] F. Irgens, *Continuum Mechanics*. Springer, 2008, isbn: 978-3-540-74297-5.
- [59] K. Suwa, T. Saitoh, Y. Takehara, M. Sano, M. Nobuhara, M. Saotome, T. Urushida, H. Katoh, H. Satoh, M. Sugiyama, T. Wakayama, M. Alley, H. Sakahara and H. Hayashi, ‘Characteristics of intra-left atrial flow dynamics and factors affecting formation of the vortex flow’, *Circulation Journal*, vol. 79, no. 1, pp. 144–152, 2014. doi: 10.1253/circj.CJ-14-0562.

Appendix A

Mesh

A.1 Mesh settings and statistics

Table A.1: Mesh settings and statistics for static and dynamic mesh performed in ANSYS Meshing. Same settings apply for all mesh cases with corresponding element size. For dynamic cases the table values correspond to the initial mesh.

Mesh case		1mm	1.5mm	2mm					
Size settings	Element size	1mm	1.5mm	2mm					
	Max size	1mm	1.5mm	2mm					
	Defeature size	0.5mm	0.5mm	0.1mm					
	Growth rate		1.05						
Quality	Target skewness		0.7						
	Smoothing		High						
Inflation	Inflation option	Smooth transition							
	Transition ratio	0.272 (standard)							
	Maximum layers	5							
	Growth rate	1.01							
		Static T01			Static T12			Dynamic	
Mesh case		1mm	1.5mm	2mm	1mm	1.5mm	2mm	1.5mm	2mm
Statistics	Nodes	351.406	128.092	64.834	419.248	175.218	85.441	36.540	17.885
	Cells	1.270.020	402.588	183.518	1.547.748	546.379	243.310	119.010	51.644
	Avg. Orth. quality	0.79343	0.74357	0.72209	0.77640	0.72443	0.71267	0.73509	0.76591
	Min. Orth. quality	0.15126	0.17973	0.14242	0.22030	0.21452	0.08884	0.10274	0.04368
	Avg. Cell ev. skew	0.20580	0.25579	0.27735	0.22289	0.27487	0.28668	0.26417	0.23336
	Max. Cell ev. skew	0.84874	0.82027	0.79592	0.77970	0.78548	0.91116	0.89726	0.95632

A.2 Dynamic mesh settings

Table A.2: Fluent dynamic mesh settings.

Dynamic	Zone	Type	Geometry definition	Meshing option	
mesh zones	Interior	Deforming	Faceted	Remeshing & smoothing	
	PV1	Deforming	Faceted (failed UDF)	Remeshing & smoothing	
	PV2	Deforming	Faceted (failed UDF)	Remeshing & smoothing	
	PV3	Deforming	Faceted (failed UDF)	Remeshing & smoothing	
	PV4	Deforming	Faceted (failed UDF)	Remeshing & smoothing	
	Outlet	Deforming	UDF	Remeshing & smoothing	
	Wall	UDF		Cell height 0.5mm	
Smoothing	Diffusion	Boundary distance	Diffusion parameter $\alpha = 2$		
Remeshing	Local cell				
	Local face				
	Parameters	Min length	0.2mm		
		Max length	3.0mm		
		Max cell skew	0.9		
		Max face skew	0.7		
		Remesh interval	1		
Mesh case		1.5mm	2mm		
Size function	Resolution	1		1	
	Variation	4.077		4.867	
	Rate	0.7		0.7	
Events	At time 0.807s	Replace mesh			
	At time 1.614s	Replace mesh			
	At time 2.421s	Replace mesh			

Appendix B

Fluent UDFs

B.1 Dynamic mesh

```
1  /*****  
2  /*LA dynamc mesh Fluent UDF */  
3  *****/  
4  /*Fluent macro library*/  
5  #include "udf.h"  
6  #include "stdio.h"  
7  #include "stdlib.h"  
8  #include "sg_mem.h"  
9  #include "dynamesh_tools.h"  
10 /*****  
11 /** Global UDF-variables **/  
12  
13 #define wall_nodes 2320 /*Number of nodes in dynamic surface  
    mesh*/  
14 #define local_intervals 25 /*Number of spline intervals*/  
15 #define line_break 26 /*Number of start and end points of  
    spline intervals*/  
16 #define TOL (1.0e-06) /*Tolerance used for identifying nodes*/  
17  
18  
19 #include /*Directory*/ /*Coefficient tables generated in  
    MATLAB*/  
20 /* All tables  
21 :  
22 :  
23 */  
24  
25 real xwall[wall_nodes]; /*Coordinates calculated from  
    coefficients*/  
26 real ywall[wall_nodes];  
27 real zwall[wall_nodes];  
28 real node_dist[wall_nodes];  
29  
30 real xf; /*Coordinates from Fluent*/  
31 real yf;  
32 real zf;  
33  
34 int fail_nodes = 0; /*Print to console variable*/  
35 int dist_node = 0; /*Print to console variable*/  
36  
37 /*****  
38  
39 **** Moving wall; start ****/  
40 DEFINE_GRID_MOTION(moving_wall, domain, dt, time, dtime)  
41 {  
42     #if !RP_HOST /*Parallelizing of the UDF*/  
43  
44     Thread *tf = DT_THREAD(dt); /*Pointer to a thread*/
```

```

45 face_t f; /*Face identifier*/
46 Node *np; /*Pointer to a node*/
47 int n; /*Node identifier*/
48 int id; /*Node ID*/
49 int i; /*Iterating integer*/
50 int coff_line; /*Line of coordinate coefficients*/
51 real tt = 0.0; /*Local time used in spline intervals*/
52 int s; /*Nr of completed heart cycles*/
53 real min_dist; /*Minmum distance between nodes*/
54
55 SET_DEFORMING_THREAD_FLAG (THREAD_TO(tf));
56 tt = CURRENT_TIME; /*Simulation time from Fluent*/
57
58 if (tt >= pbreak[line_break-1]){/*Determine local heart cycle
59     time*/
60     s = (int)floor(tt/pbreak[line_break-1]);
61     tt = tt - s*pbreak[line_break-1];
62 }
63 for (i=0; i<line_break-1; i++){ /*Determine coefficient line*/
64     if ((tt >= pbreak[i]) && (tt < pbreak[i+1])){
65         coff_line = i;
66     }
67 }
68 tt = tt - pbreak[coff_line]; /*Determine local spline time*/
69
70 for (i=0; i<wall_nodes; i++){ /*Calculating new coordinates
71     */
72     xwall[i]=((Ax_wall[coff_line][i]*tt + Bx_wall[coff_line][i])*
73         tt + Cx_wall[coff_line][i])*tt + Dx_wall[coff_line][i];
74     ywall[i]=((Ay_wall[coff_line][i]*tt + By_wall[coff_line][i])*
75         tt + Cy_wall[coff_line][i])*tt + Dy_wall[coff_line][i];
76     zwall[i]=((Az_wall[coff_line][i]*tt + Bz_wall[coff_line][i])*
77         tt + Cz_wall[coff_line][i])*tt + Dz_wall[coff_line][i];
78 }
79 begin_f_loop(f,tf){ /*Start of primary loop. Loop over
80     faces*/
81     if PRINCIPAL_FACE_P(f,tf){ /*If face belongs to partition*/
82         f_node_loop(f,tf,n){ /*Loop over nodes in selected face*/
83             np = F_NODE(f,tf,n);/*Node selected*/
84             if (NODE_POS_NEED_UPDATE(np)){ /*If node has not been
85                 updated*/
86                 NODE_POS_UPDATED(np); /*Node set as updated*/
87                 xf = NODE_X(np); /*Fluent node position*/
88                 yf = NODE_Y(np);
89                 zf = NODE_Z(np);
90                 id = -1;
91                 for (i=0; i<wall_nodes; i++){/*Identify node based on
92                     difference*/
93                     if ((fabs(xf-xwall[i]) <= TOL) && (fabs(yf-ywall[i])
94                         <= TOL) && (fabs(zf-zwall[i]) <= TOL)){
95                         id = i;
96                     }
97                 }
98             }
99             if (id == -1){/*Increase tolerance if node can not be
100                 found*/
101                 for (i=0; i<wall_nodes; i++){
102                     if ((fabs(xf-xwall[i]) <= 10.0*TOL) && (fabs(yf-
103                         ywall[i]) <= 10.0*TOL) && (fabs(zf-zwall[i]) <=
104                         10.0*TOL)){
105                         id = i;
106                     }
107                 }
108             }
109             if (id == -1){/*Increase tolerance if node can not be

```

```

    found*/
97     for (i=0; i<wall_nodes; i++){
98         if ((fabs(xf-xwall[i]) <= 100.0*TOL) && (fabs(yf-
            ywall[i]) <= 100.0*TOL) && (fabs(zf-zwall[i]) <=
                100.0*TOL)){
99             id = i;
100         }
101     }
102 }
103 if (id == -1){/*Find node based on minimum distance*/
104     min_dist = 1;
105     for (i=0; i<wall_nodes; i++){
106         node_dist[i] = sqrt(pow((xf-xwall[i]),2) + pow((yf-
            ywall[i]),2) + pow((zf-zwall[i]),2));
107         if (node_dist[i] < min_dist){
108             min_dist = node_dist[i];
109             id = i;
110         }
111     }
112     dist_node++;
113 }
114 if (id == -1){ /*Node update failed*/
115     fail_nodes++;
116 }
117 else { /*Update node position to the next time step*/
118     NODE_X(np) = xwall[id];
119     NODE_Y(np) = ywall[id];
120     NODE_Z(np) = zwall[id];
121 }
122 }
123 }
124 }
125 }
126 end_f_loop(f, tf); /*End of primary loop*/
127 if (fail_nodes > 0){
128     Message("\n Failed to update %d nodes",fail_nodes);
129     fail_nodes = 0;
130 }
131 if (dist_node > 0){
132     Message("\n Min distance used on %d nodes",dist_node);
133     dist_node = 0;
134 }
135 #endif /*Parallelizing*/
136 }
137
138
139 /**** Moving wall; end ****/
140
141 /*****
142
143 /** Moving MV plane; start **/
144 DEFINE_GEOM(MV_plane, domain_mv, dt_mv, position)
145 {
146     #if !RP_HOST /*Parallelizing*/
147
148     int j; /*Iterating integer*/
149     int coff_line_mv; /*Line of coordinate coefficients*/
150     real tmv; /*Local time used in spline intervals*/
151     int r; /*Nr of completed heart cycles*/
152     real d; /*Node distance from plane*/
153     real mv_p[3]; /*Arbitrary plane point*/
154     real mv_n[3]; /*Plane normal*/
155
156     tmv = CURRENT_TIME; /*Simulation time from Fluent*/
157
158     if (tmv >= pbreak[line_break-1]){/*Determine local heart

```

```

159     cycle time*/
160     r = (int)floor(tmv/pbreak[line_break-1]);
161     tmv = tmv - r*pbreak[line_break-1];
162 }
163 for (j=0; j<line_break-1; j++){ /*Determine coefficient line*/
164     if ((tmv >= pbreak[j]) && (tmv < pbreak[j+1])){
165         coff_line_mv = j;
166     }
167 }
168 tmv = tmv - pbreak[coff_line_mv];/*Determine local spline time
169 */
170 /*Calculate plane arbitrary point*/
171 mv_p[0] = ((Ax_mv_p[coff_line_mv]*tmv + Bx_mv_p[coff_line_mv])*
172     tmv + Cx_mv_p[coff_line_mv])*tmv + Dx_mv_p[coff_line_mv];
173 mv_p[1] = ((Ay_mv_p[coff_line_mv]*tmv + By_mv_p[coff_line_mv])*
174     tmv + Cy_mv_p[coff_line_mv])*tmv + Dy_mv_p[coff_line_mv];
175 mv_p[2] = ((Az_mv_p[coff_line_mv]*tmv + Bz_mv_p[coff_line_mv])*
176     tmv + Cz_mv_p[coff_line_mv])*tmv + Dz_mv_p[coff_line_mv];
177 /*Calculate plane normal*/
178 mv_n[0] = ((Ax_mv_n[coff_line_mv]*tmv + Bx_mv_n[coff_line_mv])*
179     tmv + Cx_mv_n[coff_line_mv])*tmv + Dx_mv_n[coff_line_mv];
180 mv_n[1] = ((Ay_mv_n[coff_line_mv]*tmv + By_mv_n[coff_line_mv])*
181     tmv + Cy_mv_n[coff_line_mv])*tmv + Dy_mv_n[coff_line_mv];
182 mv_n[2] = ((Az_mv_n[coff_line_mv]*tmv + Bz_mv_n[coff_line_mv])*
183     tmv + Cz_mv_n[coff_line_mv])*tmv + Dz_mv_n[coff_line_mv];
184 /*Distance from plane*/
185 d = mv_n[0]*(mv_p[0]-position[0]) + mv_n[1]*(mv_p[1]-position
186     [1]) + mv_n[2]*(mv_p[2]-position[2]);
187 /*New position*/
188 position[0] = position[0] + mv_n[0]*d;
189 position[1] = position[1] + mv_n[1]*d;
190 position[2] = position[2] + mv_n[2]*d;
191 #endif
192 }
193 /** Moving MV plane; end **/
194 /*PV1 - PV4 planes are calculated identical to MV plane with
195     other coefficients*/
196 /*Note:
197     MV - outlet
198     PV1 - sinister superior PV inlet
199     PV2 - sinister inferior PV inlet
200     PV3 - dexter inferior PV inlet
201     PV4 - dexter superior PV inlet*/

```

B.2 Polynomial coefficients

```

1  /******
2  /*Cubic spline interpolation coefficients generated in MATLAB*/
3  /******
4  /*Wall coefficients*/
5  real Ax_wall[local_intervals][wall_nodes] = {
6  {26.12690648,26.55203518,/*... All wall nodes*/},
7  {}},
8  /*All spline intervals
9  :
10 :
11 */
12 {}
13 };
14 /*Bx_wall, Cx_wall, Dx_wall defined the same*/
15 /*y and z coefficients defined the same*/

```

```

16 /*****
17 /*Start and end points of spline intervals*/
18 real pbreak[line_break] = {
19 0.00000000,0.03228000,/*...*/,0.80700000
20 };
21 /*****
22 /*Plane coefficients*/
23 real Ax_mv_n[local_intervals] = {
24 {-321.93025271},
25 {64.37769294},
26 {}},
27 /*All spline intervals
28 .
29 */
30 */
31 {}
32 };
33 /*Bx_mv_n, Cx_mv_n, Dx_mv_n defined the same*/
34 /*y and z coefficients defined the same*/
35 /*Arbitrary point defined the same*/
36 /*****
37 /*PV1 - PV4 plane coefficients defined the same*/
38 /*****

```

B.3 Mass flow inlet

```

1 /*****
2 /*---- Inflow PVs Fluent UDF ----*/
3 /*****
4 /*Fluent macro library*/
5 #include "udf.h"
6 #include "stdio.h"
7 #include "stdlib.h"
8 #include "sg_mem.h"
9 #include "dynamesh_tools.h"
10
11 /***** Global UDF-variables *****/
12
13 #define flow_intervals 30 /*Nr of spline intervals flow data*/
14 #define flow_break 31 /*Nr of start and end points flow splines
15 */
16 #define volume_intervals 25 /*Nr of spline intervals dv/dt*/
17 #define volume_break 26 /*Nr of start and end points dv/dt
18 splines*/
19 #define joint_intervals 2 /*Nr of intersection spline intervals*/
20 #define rho 1060 /*Density*/
21
22 #include "/*Directory*/" /*Coefficient tables generated from
23 MATLAB*/
24 /* All tables
25 :
26 :
27 */
28
29 /**Mass flow profile PV1; start**/
30 DEFINE_PROFILE(massflow_p1,t,i)
31 {
32 #if !RP_HOST /*Parallelizing UDF*/
33 face_t f; /*Face identifier*/
34 int j; /*Iterating integer*/
35 int coff_vein; /*Line of flow coefficients*/
36 int coff_volume; /*Line of dv/dt coefficients*/
37 int s; /*Nr of completed cycles*/
38 int r; /*Nr of completed cycles*/
39 real p1; /*PV1 flow*/
40 real p2; /*PV2 flow*/

```

```

38  real p3;          /*PV3 flow*/
39  real p4;          /*PV4 flow*/
40  real dv;          /*dv/dt*/
41
42  real tf = CURRENT_TIME; /*Simulation time from Fluent*/
43  real tv = CURRENT_TIME; /*Simulation time from Fluent*/
44
45  if (tf >= fbreak[flow_break-1]){/*Determine local heart cycle
    time*/
46      s = floor(tf/fbreak[flow_break-1]);
47      tf = tf - s*fbreak[flow_break-1];
48  }
49  for (j=0; j<flow_break-1; j++){/*Determine flow coefficient
    line*/
50      if ((tf >= fbreak[j]) && (tf < fbreak[j+1])){
51          coff_vein = j;
52      }
53  }
54  tf = tf - fbreak[coff_vein]; /*Determine local flow spline
    time*/
55
56  if (tv >= vbreak[volume_break-1]){/*Determine local heart cycle
    time*/
57      r = floor(tv/vbreak[volume_break-1]);
58      tv = tv - r*vbreak[volume_break-1];
59  }
60  for (j=0; j<volume_break-1; j++){/*Determine dv/dt coefficient
    line*/
61      if ((tv >= vbreak[j]) && (tv < vbreak[j+1])){
62          coff_volume = j;
63      }
64  }
65  tv = tv - vbreak[coff_volume]; /*Determine local dv/dt spline
    time*/
66  /*Calculation of flow profile*/
67  if (coff_volume == 0) { /*If second half first intersection*/
68      coff_volume = 1;
69      p1 = ((A1_J1[coff_volume]*tv + B1_J1[coff_volume])*tv + C1_J1
    [coff_volume])*tv + D1_J1[coff_volume];
70  }
71  else if (coff_volume == 10) { /*If first half second
    intersection*/
72      coff_volume = 0;
73      p1 = ((A1_J2[coff_volume]*tv + B1_J2[coff_volume])*tv + C1_J2
    [coff_volume])*tv + D1_J2[coff_volume];
74  }
75  else if (coff_vein == 10) { /*If second half second intersection
    */
76      coff_vein = 1;
77      p1 = ((A1_J2[coff_vein]*tv + B1_J2[coff_vein])*tv + C1_J2[
    coff_vein])*tv + D1_J2[coff_vein];
78  }
79  else if (coff_vein == 29) { /*If second half first intersection
    */
80      coff_vein = 0;
81      p1 = ((A1_J1[coff_vein]*tf + B1_J1[coff_vein])*tf + C1_J1[
    coff_vein])*tf + D1_J1[coff_vein];
82  }
83  else if (coff_volume < 10) { /*If systole*/
84      p1 = ((A1[coff_vein]*tf + B1[coff_vein])*tf + C1[coff_vein])*
    tf + D1[coff_vein];
85      p2 = ((A2[coff_vein]*tf + B2[coff_vein])*tf + C2[coff_vein])*
    tf + D2[coff_vein];
86      p3 = ((A3[coff_vein]*tf + B3[coff_vein])*tf + C3[coff_vein])*

```



```
      tf + D3[coff_vein];
87   p4 = ((A4[coff_vein]*tf + B4[coff_vein])*tf + C4[coff_vein])*
      tf + D4[coff_vein];
88   dv = (AV[coff_volume]*tv + BV[coff_volume])*tv + CV[
      coff_volume];
89   p1 = p1/(p1 + p2 + p3 + p4)*dv*rho;
90   }
91   else { /*If diastole*/
92     p1 = ((A1[coff_vein]*tf + B1[coff_vein])*tf + C1[coff_vein])*
      tf + D1[coff_vein];
93   }
94   /*Update flow profile to Fluent*/
95   begin_f_loop(f,t)
96   {
97     F_PROFILE(f,t,i) = p1;
98   }
99   end_f_loop(f,t);
100  #endif /*Parallelizing*/
101 }
102 /**Mass flow profile PV1; end**/
103 /*PV2 - PV4 flow profile similarly calculated*/
```


Appendix C

MATLAB Scripts

C.1 Adjust nodes

```
1 clear; close all; clc;
2
3 base_stl = stlread('new_stl\T01.stl'); %Initial STL used for
   identification
4 step_stl = stlread('new_stl\T03.stl'); %STL to be translated
5
6 a = base_stl.Points;
7 b = base_stl.ConnectivityList;
8 %Inlet and outlet nodes from Fluent used for identification in
   STL.
9 %Converted to [mm]
10 wall_import = readmatrix('wall_nodes.txt');
11 wall_nodes = wall_import(:,2:4)*1000;
12 mv_import = readmatrix('MV_plane_nodes.txt');
13 mv_nodes = mv_import(:,2:4)*1000;
14 PV1_import = readmatrix('PV1_plane_nodes.txt');
15 PV1_nodes = PV1_import(:,2:4)*1000;
16 PV2_import = readmatrix('PV2_plane_nodes.txt');
17 PV2_nodes = PV2_import(:,2:4)*1000;
18 PV3_import = readmatrix('PV3_plane_nodes.txt');
19 PV3_nodes = PV3_import(:,2:4)*1000;
20 PV4_import = readmatrix('PV4_plane_nodes.txt');
21 PV4_nodes = PV4_import(:,2:4)*1000;
22
23 %Create identificcacion array and potential missing nodes
24 %Function identify_nodes() at bottom of script
25 [MVx,~] = identify_nodes(mv_nodes,a);
26 [PV1x,~] = identify_nodes(PV1_nodes,a);
27 [PV2x,~] = identify_nodes(PV2_nodes,a);
28 [PV3x,~] = identify_nodes(PV3_nodes,a);
29 [PV4x,~] = identify_nodes(PV4_nodes,a);
30
31 %Find inlet and outlet nodes to be translated
32 c = step_stl.Points;
33 MVpoints = c(MVx,:);
34 PV1points = c(PV1x,:);
35 PV2points = c(PV2x,:);
36 PV3points = c(PV3x,:);
37 PV4points = c(PV4x,:);
38
39 %Fit plane through nodes to be translated
40 %Function plane_fit() at bottom of script
41 [MVp,MVn,MVv] = plane_fit(MVpoints);
42 [PV1p,PV1n,PV1v] = plane_fit(PV1points);
43 [PV2p,PV2n,PV2v] = plane_fit(PV2points);
44 [PV3p,PV3n,PV3v] = plane_fit(PV3points);
```

```

45 [PV4p,PV4n,PV4v] = plane_fit(PV4points);
46
47 %Translate nodes
48 %Function translate_node() at bottom of script
49 new_MV_points = translate_node(MVpoints,MVp,MVn);
50 new_PV1_points = translate_node(PV1points,PV1p,PV1n);
51 new_PV2_points = translate_node(PV2points,PV2p,PV2n);
52 new_PV3_points = translate_node(PV3points,PV3p,PV3n);
53 new_PV4_points = translate_node(PV4points,PV4p,PV4n);
54
55 %Update node positions
56 new_stl_points = c;
57 new_stl_con = b;
58 new_stl_points(MVx,:) = new_MV_points;
59 new_stl_points(PV1x,:) = new_PV1_points;
60 new_stl_points(PV2x,:) = new_PV2_points;
61 new_stl_points(PV3x,:) = new_PV3_points;
62 new_stl_points(PV4x,:) = new_PV4_points;
63
64 %Write new STL file
65 stlwrite(name,new_stl_con,new_stl_points);
66 %STL is first checked in SpaceClaim for geometrical errors
67 store = input('Save stl? y/n\n','s');
68 if store == 'y'
69     %Directory to which the STL is to be saved
70     copyfile(name,"...");
71 end
72 delete(name); %Delete STL in MATLAB directory
73
74 %Fit a plane through a point cloud
75 %p: mean plane origo. n: plane normal column vector. v: plane
    column vectors
76 function [p,n,v] = plane_fit(points)
77     p = mean(points);
78     a = points - p;
79     [eig_v,~] = eig(a'*a);
80     n = eig_v(:,1);
81     v = eig_v(:,2:end);
82 end
83
84 %Translate points along normal vector of plane to the furthest
    point
85 %(positive)
86 %points: points to be translated. p0: point on plane. n: normal
    column
87 %vector of plane
88 %new_points: translated points
89 function [new_points] = translate_node(points,p0,n),id)
90     new_points = zeros(size(points));
91     dl = zeros(1,length(points));
92     for j = 1:length(points)
93         dl(j) = sum(n'.*(p0-points(j,:)));
94     end
95     p_new = p0 + max(dl)*n';
96     for k = 1:length(points)
97         dl(k) = sum(n'.*(p_new-points(k,:)));
98         new_points(k,:) = points(k,:) + n'*dl(k);
99     end
100 end
101
102 %Identify corresponding surface nodes of STL and Fluent
103 %id: column vector numbering the node rows of STL matching Fluent
    nodes
104 %error_node: Fluent nodes which could not be found
105 function [id,error_node] = identify_nodes(nodes_f,nodes)

```

```

106     id = [];
107     error_node = [];
108     count = 0;
109     for i = 1:length(nodes_f)
110         [tf, index] = ismembertol(nodes_f(i,:),nodes,0.00001,'
            ByRows',true,'DataScale',1);
111         if tf==0
112             count = count + 1;
113             error_node(count) = i;
114         else
115             id(i-count) = index;
116         end
117     end
118 end

```

C.2 Cubic spline interpolation

```

1  clear; close all; clc;
2
3  time_interval = 807/1000;    %Cycle time in [s]
4  dt = time_interval/25;      %Time step size
5  t = 0:dt:time_interval;     %Time vector
6  t0_stl = stlread('new_stl\T01.stl'); %Initial time step
7  %Inlet and outlet nodes from Fluent used for identification in
   STL.
8  %Converted to [mm]
9  wall_import = readmatrix('wall_nodes.txt');
10 wall_nodes = wall_import(:,2:4)*1000;
11 mv_import = readmatrix('MV_plane_nodes.txt');
12 mv_nodes = mv_import(:,2:4)*1000;
13 PV1_import = readmatrix('PV1_plane_nodes.txt');
14 PV1_nodes = PV1_import(:,2:4)*1000;
15 PV2_import = readmatrix('PV2_plane_nodes.txt');
16 PV2_nodes = PV2_import(:,2:4)*1000;
17 PV3_import = readmatrix('PV3_plane_nodes.txt');
18 PV3_nodes = PV3_import(:,2:4)*1000;
19 PV4_import = readmatrix('PV4_plane_nodes.txt');
20 PV4_nodes = PV4_import(:,2:4)*1000;
21
22 %Create identificcation array and potential missing nodes
23 %Function identify_nodes() at bottom of script
24 [id_wall,e_wall] = identify_nodes(wall_nodes,t0_stl.Points);
25 [id_mv,e_mv] = identify_nodes(mv_nodes,t0_stl.Points);
26 [id_PV1,e_pv1] = identify_nodes(PV1_nodes,t0_stl.Points);
27 [id_PV2,e_pv2] = identify_nodes(PV2_nodes,t0_stl.Points);
28 [id_PV3,e_pv3] = identify_nodes(PV3_nodes,t0_stl.Points);
29 [id_PV4,e_pv4] = identify_nodes(PV4_nodes,t0_stl.Points);
30
31 %Directory of the 25 stl files
32 old_dir = dir('new_stl');
33 %Directories start with "." ".."
34 base_stl = stlread([old_dir(3).folder filesep old_dir(3).name]);
35 %Create initial point arrays
36 x_points_wall = zeros(length(id_wall),length(old_dir)-2);
37 y_points_wall = x_points_wall;
38 z_points_wall = x_points_wall;
39 x_points_mv = zeros(length(id_mv),length(old_dir)-2);
40 y_points_mv = x_points_mv;
41 z_points_mv = x_points_mv;
42 %Same for PV1-PV4
43
44 %Aquire points. Converted to [m]
45 for j = 3:length(old_dir)
46     step_stl = stlread([old_dir(j).folder filesep old_dir(j).name

```

```

    ]);
47     for k = 1:length(id_wall)
48         x_points_wall(k,j-2) = step_stl.Points(id_wall(k),1)
           /1000;
49         y_points_wall(k,j-2) = step_stl.Points(id_wall(k),2)
           /1000;
50         z_points_wall(k,j-2) = step_stl.Points(id_wall(k),3)
           /1000;
51     end
52     for l = 1:length(id_mv)
53         x_points_mv(l,j-2) = step_stl.Points(id_mv(l),1)/1000;
54         y_points_mv(l,j-2) = step_stl.Points(id_mv(l),2)/1000;
55         z_points_mv(l,j-2) = step_stl.Points(id_mv(l),3)/1000;
56     end
57     %Same for PV1-PV4
58 end
59
60 %Extend point array for spline interpolation. First time step
    equals last.
61 %Extra points at start and end to make the cycle continuous for
    spline
62 %interpolation. See documentation of spline() for more
63 x_points_wall = [x_points_wall(:,end) x_points_wall x_points_wall
    (:,1) x_points_wall(:,2)];
64 y_points_wall = [y_points_wall(:,end) y_points_wall y_points_wall
    (:,1) y_points_wall(:,2)];
65 z_points_wall = [z_points_wall(:,end) z_points_wall z_points_wall
    (:,1) z_points_wall(:,2)];
66 x_points_mv = [x_points_mv(:,end) x_points_mv x_points_mv(:,1)
    x_points_mv(:,2)];
67 y_points_mv = [y_points_mv(:,end) y_points_mv y_points_mv(:,1)
    y_points_mv(:,2)];
68 z_points_mv = [z_points_mv(:,end) z_points_mv z_points_mv(:,1)
    z_points_mv(:,2)];
69 %Same for PV1-PV4
70
71 %Create initial wall polynomial array
72 pp_wall = spline(t,x_points_wall(1,:));
73 xcoeff_wall = zeros(pp_wall.pieces,pp_wall.order,length(
    x_points_wall));
74 ycoeff_wall = xcoeff_wall;
75 zcoeff_wall = xcoeff_wall;
76 xbreak_wall = zeros(length(x_points_wall),length(pp_wall.breaks))
    ;
77 ybreak_wall = xbreak_wall;
78 zbreak_wall = xbreak_wall;
79
80 %Wall spline interpolation. Save Spline coefficients and breaks
81 for i = 1:length(x_points_wall)
82     ppx_wall = spline(t,x_points_wall(i,:));
83     ppy_wall = spline(t,y_points_wall(i,:));
84     ppz_wall = spline(t,z_points_wall(i,:));
85     xcoeff_wall(:, :, i) = ppx_wall.coefs;
86     ycoeff_wall(:, :, i) = ppy_wall.coefs;
87     zcoeff_wall(:, :, i) = ppz_wall.coefs;
88     xbreak_wall(i, :) = ppx_wall.breaks;
89     ybreak_wall(i, :) = ppy_wall.breaks;
90     zbreak_wall(i, :) = ppz_wall.breaks;
91 end
92
93 %Check if spline breaks are consistent
94 if sum(abs(mean(xbreak_wall)-xbreak_wall(1,:))<=1.0e-10)==length
    (pp_wall.breaks) && sum(abs(mean(ybreak_wall)-xbreak_wall
    (1,:))<=1.0e-10)==length(pp_wall.breaks) && sum(abs(mean(

```

```

        zbreak_wall)-xbreak_wall(1,:))<=1.0e-10)==length(pp_wall.
        breaks)
95     disp('Wall spline-breaks are good');
96     breaks = double(xbreak_wall(1,:));
97 else
98     disp('Wall spline-breaks are inconsistent');
99 end
100
101 %Create initial plane polynomial array
102 mv_n = zeros(length(t)+2,3); %Normal vector
103 mv_p = mv_n; %Arbitrary point
104 %Same for PV1-PV4
105
106 %Create normal vector and arbitrary point.
107 for i = 1:length(t)+2
108     mv_points = [x_points_mv(:,i) y_points_mv(:,i) z_points_mv
109                 (:,i)];
110     %Function plane_fit() at bottom of script
111     [mv_p(i,:),mv_n(i,:),~] = plane_fit(mv_points);
112     %Keep sign of normal vector constant. Issue for spline
113     interpolation
114     if sum(mv_n(i,:).*mv_n(1,:)) < 0
115         mv_n(i,:) = mv_n(i,~)*-1;
116     end
117     %Same for PV1-PV4
118 end
119 %Plane spline interpolation
120 ppx_mv_p = spline(t,mv_p(:,1));
121 ppy_mv_p = spline(t,mv_p(:,2));
122 ppz_mv_p = spline(t,mv_p(:,3));
123 ppx_mv_n = spline(t,mv_n(:,1));
124 ppy_mv_n = spline(t,mv_n(:,2));
125 ppz_mv_n = spline(t,mv_n(:,3));
126 %Same for PV1-PV4
127
128 %Identify corresponding surface nodes of STL and Fluent
129 %id: column vector numbering the node rows of STL matching Fluent
130 nodes
131 %error_node: Fluent nodes which could not be found
132 function [id,error_node] = identify_nodes(nodes_f,nodes)
133     id = [];
134     error_node = [];
135     count = 0;
136     for i = 1:length(nodes_f)
137         [tf, index] = ismembertol(nodes_f(i,:),nodes,0.00001,'
138                                 ByRows',true,'DataScale',1);
139         if tf==0
140             count = count + 1;
141             error_node(count) = i;
142         else
143             id(i-count) = index;
144         end
145     end
146 end
147
148 %Fit a plane through a point cloud
149 %p: mean plane origo. n: plane normal column vector. v: plane
150 column vectors
151 function [p,n,v] = plane_fit(points)
152     p = mean(points);
153     a = points - p;
154     [eig_v,~] = eig(a'*a);
155     n = eig_v(:,1);
156     v = eig_v(:,2:end);
157 end

```

C.3 PV flow adjustment and spline interpolation

```

1 clear; close all; clc;
2
3 rho = 1060; %density kg/m^3
4 %PV flow ml/s from Dahl et al.
5 flow = [22.850872 20.351944 10.106912 14.505304;
6         33.362198 30.106018 20.104597 27.708527;
7         38.89637 34.82539 27.077345 32.866993;
8         39.641521 34.974007 29.373987 32.103298;
9         40.905342 33.252846 33.791016 30.280092;
10        47.189346 37.588131 39.350334 30.935957;
11        55.044125 45.96183 41.863018 31.877893;
12        57.153915 54.021072 44.212341 34.26437;
13        51.829662 55.13166 40.167721 32.911942;
14        40.782383 49.101547 33.995121 29.083527;
15        30.241171 43.277206 27.442759 24.832176;
16        27.083237 37.170723 20.577187 24.161554;
17        32.685822 38.252861 21.121267 27.752604;
18        41.646534 43.601772 26.883728 35.366993;
19        46.41629 49.216599 34.420692 43.100307;
20        44.165356 46.224754 36.87587 42.930557;
21        36.340103 38.188595 33.938335 36.583332;
22        27.005463 27.864504 28.219156 27.116222;
23        18.914301 19.456736 20.931267 20.663195;
24        13.145369 13.848521 14.137669 12.94184;
25        9.313755 11.024884 9.520514 8.611593;
26        6.957322 8.492123 6.586359 6.343171;
27        6.057343 7.382534 4.338543 5.091724;
28        5.509327 6.090135 3.405178 4.842496;
29        4.833489 3.580832 2.636014 3.717978;
30        2.054881 1.269314 0.432203 -0.834587;
31        -4.658498 -3.746172 -4.709261 -5.353395;
32        -9.014003 -8.604179 -8.402815 -7.209201;
33        -3.81757 -4.651238 -4.394482 -3.267457;
34        6.686914 6.848329 2.719923 3.597126];
35 old_t = 0:35.233333:1021.766663; %Old time from Dahl et al. ms
36 %Segmented volume mm^3
37 volume = [40846.4902 41968.0592 44287.5408 47160.2047
38           53509.6117 58980.8392 64237.2513 66555.4652 70765.2976
39           73666.7016 77801.8656 79362.5865 78038.8498 72585.0907
40           67651.9049 60470.2021 57050.5042 56934.9167 61690.0796
41           62451.2962 63084.0581 61156.8794 55944.2817 45132.4239
42           40348.2975];
43 %Volume vector for spline interpolation. m^3
44 vol_spline = [volume(end) volume volume(1) volume(2)]*1e-9;
45 %Old time vector for spline interpolation. s
46 old_t = [old_t old_t(end)+35.233333]/1000;
47 %New time vector. s
48 interval = 0.807;
49 new_t = 0:interval/25:interval;
50 %Adjust old time intervals to new intervals
51 old_dia = 11;
52 new_dia = 12;
53 t_old_dia = old_t(11:end);
54 t_old_sys = old_t(1:11);
55 t_new_dia = new_t(12:end);
56 t_new_sys = new_t(1:12);
57 adjust_dt_dia = (t_new_dia(end)-t_new_dia(1))/(length(t_old_dia)
58               -1);
59 adjust_dt_sys = (t_new_sys(end)-t_new_sys(1))/(length(t_old_sys)
60               -1);
61 adjust_t_dia = t_new_dia(1):adjust_dt_dia:t_new_dia(end);
62 adjust_t_sys = t_new_sys(1):adjust_dt_sys:t_new_sys(end);
63 adjust_t = [adjust_t_sys adjust_t_dia(2:end)];
64
65 pp_vol = spline(new_t, vol_spline); %Spline of volume

```



```

59 pp_dv = fnder(pp_vol,1); %Spline of dV/dt
60 %Mass flow for each PV. kg/s
61 mass_flow = flow'*rho/1e06;
62 p1_old = mass_flow(3,:);
63 p2_old = mass_flow(1,:);
64 p3_old = mass_flow(2,:);
65 p4_old = mass_flow(4,:);
66 %Spline of PV flow
67 p1_spline_old = [p1_old(end) p1_old p1_old(1) p1_old(2)];
68 p2_spline_old = [p2_old(end) p2_old p2_old(1) p2_old(2)];
69 p3_spline_old = [p3_old(end) p3_old p3_old(1) p3_old(2)];
70 p4_spline_old = [p4_old(end) p4_old p4_old(1) p4_old(2)];
71 pp1 = spline(adjust_t,p1_spline_old);
72 pp2 = spline(adjust_t,p2_spline_old);
73 pp3 = spline(adjust_t,p3_spline_old);
74 pp4 = spline(adjust_t,p4_spline_old);
75 pp_flow = pp1; %Spline of flow sum
76 pp_flow.coefs = pp1.coefs + pp2.coefs + pp3.coefs + pp4.coefs;
77 %Intersecting intervals of systole/diastole
78 joint1_breaks_pv = pp1.breaks([end-1 end]);
79 joint1_breaks_dv = pp_dv.breaks([1 2]);
80 joint1_breaks = [joint1_breaks_pv joint1_breaks_dv];
81 joint1_breaks_mod = [joint1_breaks_pv-interval joint1_breaks_dv
82 ];
82 joint2_breaks_pv = pp1.breaks([11 12]);
83 joint2_breaks_dv = pp_dv.breaks([11 12]);
84 joint2_breaks = [joint2_breaks_dv joint2_breaks_pv];
85 %Values at 1st interval
86 dv_values_1 = ppval(pp_dv,joint1_breaks);
87 pv1_values_1 = ppval(pp1,joint1_breaks);
88 pv2_values_1 = ppval(pp2,joint1_breaks);
89 pv3_values_1 = ppval(pp3,joint1_breaks);
90 pv4_values_1 = ppval(pp4,joint1_breaks);
91 flow_values_1 = ppval(pp_flow,joint1_breaks);
92 pv1_dv_values_1 = pv1_values_1./flow_values_1.*dv_values_1*rho;
93 pv2_dv_values_1 = pv2_values_1./flow_values_1.*dv_values_1*rho;
94 pv3_dv_values_1 = pv3_values_1./flow_values_1.*dv_values_1*rho;
95 pv4_dv_values_1 = pv4_values_1./flow_values_1.*dv_values_1*rho;
96 %Values at 2nd interval
97 dv_values_2 = ppval(pp_dv,joint2_breaks);
98 pv1_values_2 = ppval(pp1,joint2_breaks);
99 pv2_values_2 = ppval(pp2,joint2_breaks);
100 pv3_values_2 = ppval(pp3,joint2_breaks);
101 pv4_values_2 = ppval(pp4,joint2_breaks);
102 flow_values_2 = ppval(pp_flow,joint2_breaks);
103 pv1_dv_values_2 = pv1_values_2./flow_values_2.*dv_values_2*rho;
104 pv2_dv_values_2 = pv2_values_2./flow_values_2.*dv_values_2*rho;
105 pv3_dv_values_2 = pv3_values_2./flow_values_2.*dv_values_2*rho;
106 pv4_dv_values_2 = pv4_values_2./flow_values_2.*dv_values_2*rho;
107 %Differentiate to find boundary conditions for interval splines
108 ddv = fnder(pp_dv,1);
109 dpv1 = fnder(pp1,1);
110 dpv2 = fnder(pp2,1);
111 dpv3 = fnder(pp3,1);
112 dpv4 = fnder(pp4,1);
113 dflow = fnder(pp_flow,1);
114 %Differentiated values of d/dt(Qi(t)/Qtot(t)*dV/dt(t)*rho)
115 ddv_pv1_values_1 = (ppval(dpv1,joint1_breaks).*flow_values_1 -
    pv1_values_1.*ppval(dpv1,joint1_breaks))./flow_values_1.^2.*
    dv_values_1*rho + pv1_values_1./flow_values_1.*ppval(ddv,
    joint1_breaks)*rho;
116 ddv_pv2_values_1 = (ppval(dpv2,joint1_breaks).*flow_values_1 -

```

```

    pv2_values_1.*ppval(dpv2,joint1_breaks))./flow_values_1.^2.*
    dv_values_1*rho + pv2_values_1./flow_values_1.*ppval(ddv,
    joint1_breaks)*rho;
117 ddv_pv3_values_1 = (ppval(dpv3,joint1_breaks).*flow_values_1 -
    pv3_values_1.*ppval(dpv3,joint1_breaks))./flow_values_1.^2.*
    dv_values_1*rho + pv3_values_1./flow_values_1.*ppval(ddv,
    joint1_breaks)*rho;
118 ddv_pv4_values_1 = (ppval(dpv4,joint1_breaks).*flow_values_1 -
    pv4_values_1.*ppval(dpv4,joint1_breaks))./flow_values_1.^2.*
    dv_values_1*rho + pv4_values_1./flow_values_1.*ppval(ddv,
    joint1_breaks)*rho;
119 ddv_pv1_values_2 = (ppval(dpv1,joint2_breaks).*flow_values_2 -
    pv1_values_2.*ppval(dpv1,joint2_breaks))./flow_values_2.^2.*
    dv_values_2*rho + pv1_values_2./flow_values_2.*ppval(ddv,
    joint2_breaks)*rho;
120 ddv_pv2_values_2 = (ppval(dpv2,joint2_breaks).*flow_values_2 -
    pv2_values_2.*ppval(dpv2,joint2_breaks))./flow_values_2.^2.*
    dv_values_2*rho + pv2_values_2./flow_values_2.*ppval(ddv,
    joint2_breaks)*rho;
121 ddv_pv3_values_2 = (ppval(dpv3,joint2_breaks).*flow_values_2 -
    pv3_values_2.*ppval(dpv3,joint2_breaks))./flow_values_2.^2.*
    dv_values_2*rho + pv3_values_2./flow_values_2.*ppval(ddv,
    joint2_breaks)*rho;
122 ddv_pv4_values_2 = (ppval(dpv4,joint2_breaks).*flow_values_2 -
    pv4_values_2.*ppval(dpv4,joint2_breaks))./flow_values_2.^2.*
    dv_values_2*rho + pv4_values_2./flow_values_2.*ppval(ddv,
    joint2_breaks)*rho;
123 %Spline values for 1st interval
124 pv1_joint1_values = [ppval(dpv1,joint1_breaks(1)) pv1_values_1
    (1:2) pv1_dv_values_1(3:4) ddv_pv1_values_1(end)];
125 pv2_joint1_values = [ppval(dpv2,joint1_breaks(1)) pv2_values_1
    (1:2) pv2_dv_values_1(3:4) ddv_pv2_values_1(end)];
126 pv3_joint1_values = [ppval(dpv3,joint1_breaks(1)) pv3_values_1
    (1:2) pv3_dv_values_1(3:4) ddv_pv3_values_1(end)];
127 pv4_joint1_values = [ppval(dpv4,joint1_breaks(1)) pv4_values_1
    (1:2) pv4_dv_values_1(3:4) ddv_pv4_values_1(end)];
128 %Spline values for 2nd interval
129 pv1_joint2_values = [ddv_pv1_values_2(1) pv1_dv_values_2(1:2)
    pv1_values_2(3:4) ppval(dpv1,joint2_breaks(end))];
130 pv2_joint2_values = [ddv_pv2_values_2(1) pv2_dv_values_2(1:2)
    pv2_values_2(3:4) ppval(dpv2,joint2_breaks(end))];
131 pv3_joint2_values = [ddv_pv3_values_2(1) pv3_dv_values_2(1:2)
    pv3_values_2(3:4) ppval(dpv3,joint2_breaks(end))];
132 pv4_joint2_values = [ddv_pv4_values_2(1) pv4_dv_values_2(1:2)
    pv4_values_2(3:4) ppval(dpv4,joint2_breaks(end))];
133 %1st interval spline
134 pv1_joint1 = csape(joint1_breaks_mod,pv1_joint1_values,[1 1]);
135 pv2_joint1 = csape(joint1_breaks_mod,pv2_joint1_values,[1 1]);
136 pv3_joint1 = csape(joint1_breaks_mod,pv3_joint1_values,[1 1]);
137 pv4_joint1 = csape(joint1_breaks_mod,pv4_joint1_values,[1 1]);
138 %2nd interval spline
139 pv1_joint2 = csape(joint2_breaks,pv1_joint2_values,[1 1]);
140 pv2_joint2 = csape(joint2_breaks,pv2_joint2_values,[1 1]);
141 pv3_joint2 = csape(joint2_breaks,pv3_joint2_values,[1 1]);
142 pv4_joint2 = csape(joint2_breaks,pv4_joint2_values,[1 1]);
143 %Spline breaks
144 flow_breaks = pp_flow.breaks;
145 volume_breaks = pp_vol.breaks;
146
147 %Resulting spline order wrt time:
148 % 1st (0s-0.0323s) - 2nd half of joint1-spline
149 % 2nd (0.0323s-0.3228s) - PV flow spline adjusted with dV/dt

```

```
150 % 3rd (0.3228s-0.3777s) - joint2-spline
151 % 4th (0.3777s-0.7844) - Original PV flow spline
152 % 5th (0.7844s-0.807s) - 1st half of joint1-spline
```