Modelling blood flow in the circle of Willis

Table of contents

Acknowledgements

I am grateful to Dr Tabor, senior lecturer at the University of Exeter for offering me this internship and for all the advice he has given me throughout this project.

I would like to thank Dr Young and Dr Ross Cotton from Simpleware Ltd for offering me a ScanIP license and guidance on the meshing process during this project.

I am also grateful to Augusto Della Torre, visiting PhD student, for helping me setting up cases and having a greater understanding of how OpenFOAM worked.

I would also like to thank the staff of the University of Exeter library for helping me out during my research.

Abstract

The project done during the internship revolved around three main tasks.

The first one consisted in doing a survey on the various works that had been done on the modelling of blood flow in general and of the circle of Willis. The second task consisted in reconstructing and meshing the geometry of the circle of Willis, based on data collected by the Royal Devon and Exeter hospital, of three patients that suffer from a condition affecting their circle of Willis with the image-based meshing software ScanIP. The third task was to use CFD to model the blood flow in the circle of Willis with OpenFOAM and to extract data that could be useful to diagnose the patients and inform us on how blood flow behaved as a consequence of the changes in the circle of Willis induced by conditions. The modelling stressed on the outlets boundary condition by implementing the windkessel boundary condition in OpenFOAM.

Keywords: CFD, circle of Willis, patient-specific, survey, meshing, modelling, windkessel.

Introduction

Computational fluid dynamics (CFD) is a field that has largely developed in the last few years. Its applications are numerous. It is mainly used in industry and research for design and investigation purposes. This internship puts the focus on the latter aspect of CFD analysis.

Cardiovascular diseases are among the top causes of death in the world, especially in Western countries [21]. It is therefore of paramount importance to find effective ways to diagnose such diseases. The circle of Willis is a network of arteries that supplies the brain tissues in oxygen in which it is known as a fact that aneurysm, strokes or stenosis have a high risk of developing due to malformations. Studies on the circle of Willis suggest that only about forty percent of the population has a complete circle of Willis indeed.

This project is about demonstrating how CFD could be helpful in detecting and in giving clues about how risk patients' blood circulation adapts to blockages or malformations of the circle of Willis. In order to obtain insights on the circle of Willis in a non-invasive way to help doctors diagnose conditions in the circle of Willis, computer-aided medicine can play a key part. Research suggest that wall shear stress, which is difficult to measure, is a determinant factor in the development of condition such as aneurysm [13]. Other important data such as velocity and pressure are difficult to measure without sophisticated and expensive measure instruments or need a surgical operation to perform in-vivo measures.

1 Background

1.1 The circle of Willis

The circle of Willis is a ring of arteries that supplies the brain tissues in the brain thanks to three main arteries emanating from the neck which are the two carotid arteries on either sides of the neck and the basilar artery at the back of the neck. The blood from these arteries can flow through other substitute arteries that link the biggest arteries between themselves. This complex tangling of arteries is particularly useful in case a clot develops in the circle of Willis. Indeed, in case an area is blocked, the blood could be re-routed to other arteries so that the brain tissues would still be supplied in blood. In a way the circle of Willis is a redundant system of distribution of blood to the blood tissues.

Here is the classical shape of a circle of Willis:

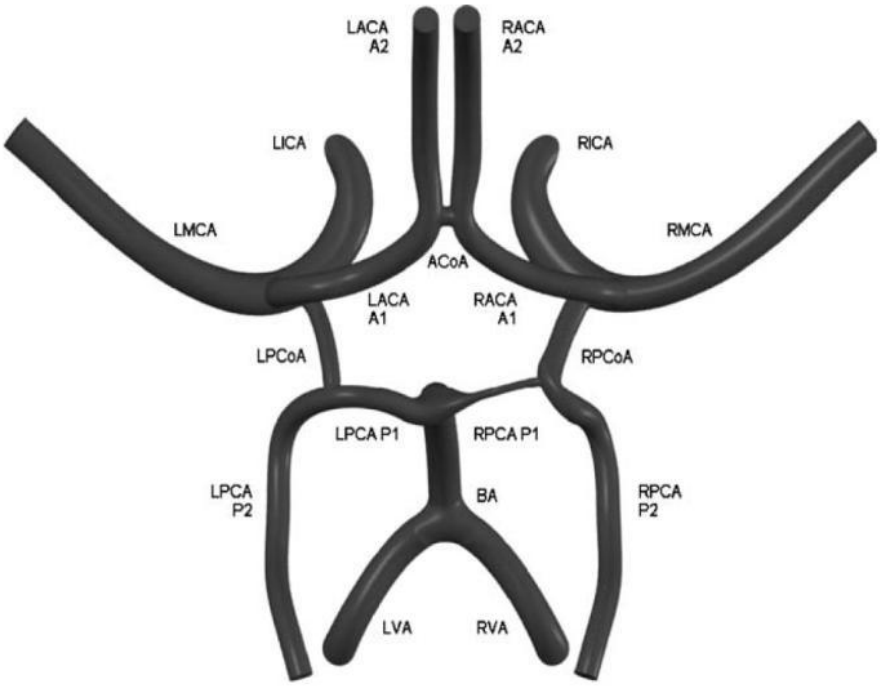

Figure 1 Complete circle of Willis [20]

List of abbreviations

BA: basilar artery
VA: vertebral artery
ICA: internal carotid artery
ACA: anterior cerebral artery
MCA: middle cerebral artery
PCoA: Posterior communicating artery
ACoA: Anterior communicating artery

The suffix L and R mean respectively left and right.

1.2 State of the art: boundary conditions (BC) in the literature

Modelling of hemodynamics can be done on different levels. Many important studies were done using 1D or 2D models. Given the fact that wall shear stress is one of the main information to be analyzed from the simulation, 3D simulations were employed. Even then the options for the simulations are numerous. Here are some of the most commonly used and most recent boundary conditions that are applied in simulations of hemodynamics of truncated arterial trees.

1.2.1 3D-1D coupling

Modelling a 3D model for the whole vascular system is highly unrealistic given the millions of blood vessels to take into account and the requirement substantial resources to compute the solution of the flow and pressure in the system. That is why 3D models consist in a majority of cases in a truncated part of the vascular system. In some work however, instead of limiting the model to a part of the vascular system, only the part of interest is represented in 3D. The remainder of the vascular system is described using a 1D model. This type of modelling is known as the 3D-1D coupling model [7].

There are variants of this model which only represent “terminal” vessels like capillaries with 1D geometries. The terminal vessels are not precisely modelled but fractals are used as an approximation to the complex network of downstream “terminal” vessels [10].

1.2.2 Lumped parameters boundary conditions

Analogy between electricity and fluid mechanics is often used to describe systems. The pressure is likened to the intensity whilst the flow rate represents an analogue of tension in electricity. Former simulations of hemodynamics in the vascular system took the approach of modelling the phenomena in blood vessels with actual electric circuits. With CFD calculations coming to the fore, this method has been less and less used but principles of it are still at the heart of the definition of some downstream boundary conditions called the lumped parameters or 0D boundary conditions. The downstream vessels’ dynamics is represented with an electric circuit from which an equation linking the pressure and the flow rate can be extracted from. This equation is then used as a boundary condition.

There are several electric circuits that have been usually used to obtain the equation for lumped parameters boundary conditions.

1.2.2.1 Resistive circuit

The easiest one to implement is the purely resistive circuit from which there is a purely linear relationship between pressure and flow rate. This boundary condition models only the peripheral resistance of the vascular bed [7].

Wall compliance is very important in the modelling of blood flows as it is understood that the elasticity of the vessels gives steady-state like property the blood flow resulting in a

nearly continuous supply of the tissues in oxygen and foodstuff [3]. This is a very important property as during the diastole period in the heart cycle no blood is pumped out of the heart.

1.2.2.2 Windkessel BC

Other well-known lumped parameters electric circuits are the windkessel ones. With this circuit, the notion of wall compliance of the downstream vessels is introduced; it is a distributed closed loop elastic system. This boundary condition has been built thanks to an analogy with compressed air chamber (windkessel meaning air chamber in german). The air chamber would contract and dilate accordingly to the incoming fluid flow thus the analogy with the vessel compliance. Thanks to the compressibility of the air chamber, the flowing fluid would be released almost steadily out of the chamber [1,3].

There are three main windkessel circuits that have been used in the literature. They are named accordingly to the number of elements in them [14,15].

To obtain the equation of the windkessel circuit a few hypothesis have been assumed to have the coupling of pressure and flow rate represented by an electric circuit.

The potential difference, rather the pressure difference p is the difference between the pressure at the outlet and a reference pressure in the venous system. The venous system, which marks the “end” of the system, has the advantage of being relatively stable in terms of pressure [2].

In the two-element windkessel boundary condition or Frank’s model the circuit is simply a parallel circuit composed of a resistance R representing the peripheral resistance to the flow and a capacitance C representing the wall compliance. The peripheral resistance is induced by the narrowing of the blood vessel, as we go from major arteries like carotids to extremely thin ones (arterioles) and capillaries.

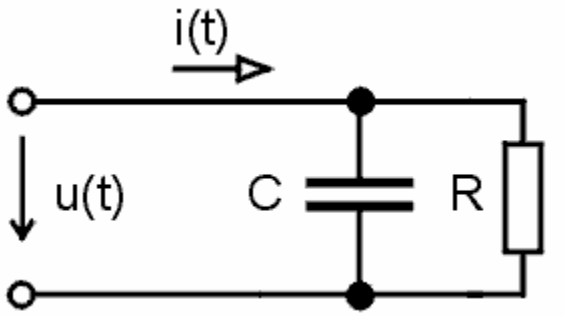

Figure 2 Two-element windkessel circuit

$$u(t) + RC\frac{du}{dt} = Ri(t)$$

In the three-element windkessel boundary condition, a resistance to the flow within the air chamber is introduced, hence the following circuit and equation.

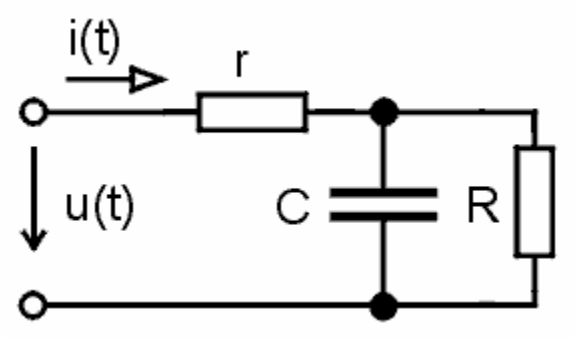

Figure 3 Three-element windkessel circuit

$$u(t) + RC\frac{du}{dt} = (r + R)i(t) + rRC\frac{di}{dt}$$

The four-element windkessel boundary condition introduces the notion of blood flow inertia with the inductance. According to the literature, it is the circuit is that reproduces the most physiological results.

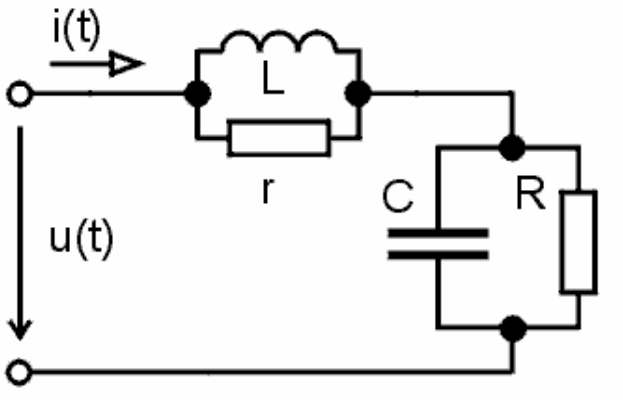

Figure 4 Four-element windkessel circuit

$$u(t) + RC\frac{du}{dt} = (r + R)i(t) + (rRC + L)\frac{di}{dt} + \mathrm{RLC}\frac{d^2i}{dt^2}$$

All in all, the windkessel BC adds compliance to the model. However, the hypothesis behind, which is that all downstream vessels contract and expand at the same time, is wrong as the vessel deformation propagate as a wave accordingly to the Womersley theory. Still, this type of boundary conditions gives better results according to various papers [5,6].

1.2.2.3 Impedance BC

The windkessel and purely resistive circuits are only part of the number of circuits that can be used as lumped parameters. Others more complex lumped parameters boundary conditions are simply described as impedance boundary conditions [8,10].

Wall compliance is one of the most important aspects of blood dynamics modelling. Moving walls boundary conditions have often been discarded in CFD studies as using this type of boundary conditions requires a fluid-structure interface that could dramatically increase the calculation times. It would indeed be necessary to calculate the displacement of the wall and solve the Navier-Stokes equation in the new fluid domain that results from the wall compliancy. Various pressure-area relationships are used for the coupling of the fluid and the structure at the interface [5, 7].

1.2.3 Immersed boundary conditions

This type of boundary condition can be an alternative to the fluid-structure interaction [9]. It will be implemented in one of the future version of OpenFOAM according to information gathered on cfd-online.com.

1.2.4 Fixed pressure

Prescribed pressure at the outlet is the most commonly used boundary conditions in hemodynamics modelling as it is easy to implement. However, a fixed pressure is far from being the most realistic boundary conditions especially in arteries where the pressure varies greatly between the systolic – when blood is ejected from the heart - and diastolic phases. A fixed pressure, usually 0 Pa, can however be physiological enough in small arteries and in veins which are known for their low pressure. A value of 5000 Pa is also used and considered as a neutral value [13].

1.2.5 Integrating the reflections at bifurcation in models

One phenomenon that is often neglected and not implemented in the modelling of blood dynamics is the effects resulting from bifurcations in the downstream vessels. Indeed, at bifurcations the flow is split in two main flows: the flow that will go through the efferent arteries and the flow that is reflected. The latter is the one that is often neglected with the usual boundary conditions. A new type of inlet described in [6] can remedy to this problem though the formulation is quite complex.

1.3 Tools

1.3.1 ScanIP

The data used to create geometry of the circle of Willis are provided by an MRI. An MRI takes slice-to-slice images of the brain by magnetic resonance. To build a 3D model of the blood vessels, the images taken by the MRI should be superposed and filtered so that only the parts that interested are shown. It would be possible to create the geometry of the circle of Willis with any CAD software but it would be extremely time-consuming.

ScanIP is a powerful tool created by Simpleware Ltd - a company which has an outpost based in Exeter - that eases the processes of creation of a geometry from an MRI along with the meshing process.

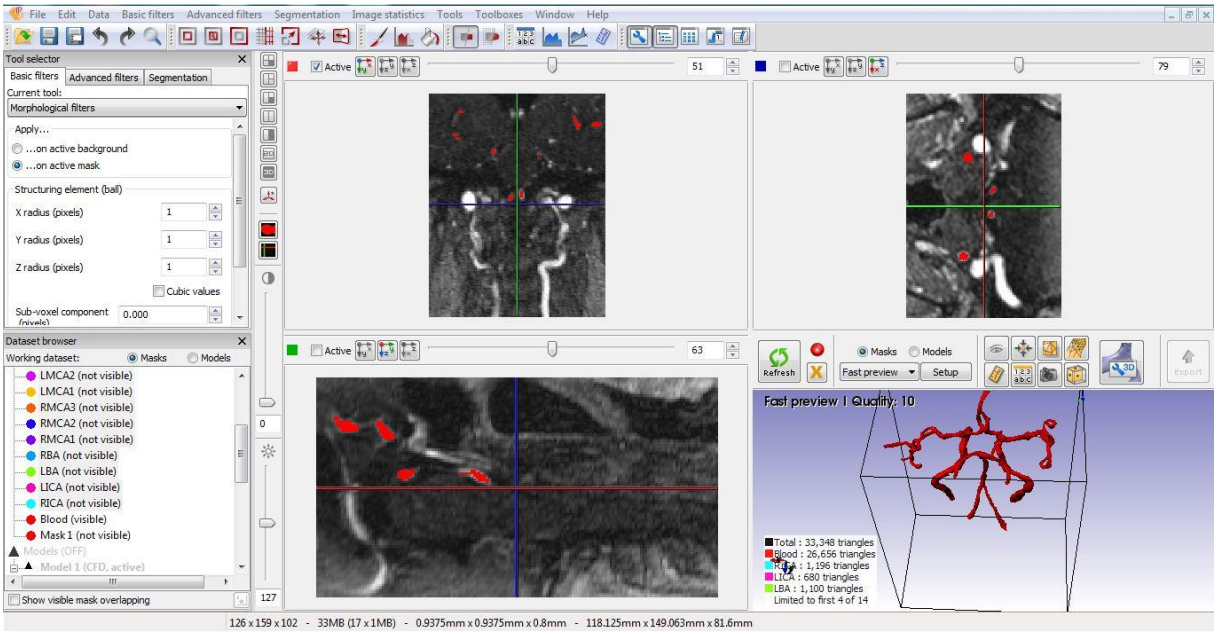

Figure 5 ScanIP screenshot

With the help of ScanIP, it is possible to create a geometry from medical images produced by MRI scans. ScanIP is particularly useful to model complex geometries.

The mesh is created accordingly to the painted voxels (3D equivalent of pixels). All the voxels are associated with a greyscale value and usually the voxels are selected by painting the voxels whose greyscale value is within a certain range.

1.3.2 OpenFOAM

The CFD code used to do all the studies in this project is the CFD toolkit called OpenFOAM.

OpenFOAM is an open source CFD toolkit that has been in constant development since its first release. It is written in C++ and uses the finite volume method. There is a growing interest in OpenFOAM from the industry as it is free and offers the possibility to create custom models unlike most commercial codes.

2 Work on geometry

2.1 Creating the geometry

Creating the geometry – or mask – and the mesh was the single most time-consuming task in this project. Although various tools are at disposal to generate the geometry from greyscale value, the best option to create it was using the paint tools available in ScanIP. The threshold technique was indeed not useful as the greyscale value of the vessels of interest varied greatly, meaning that to capture all the necessary vessels in the geometry we would need to use a wide range of greyscale values. This has the consequence of including in the geometry many unwanted parts and artefacts, see Figure 6.

Figure 6 Image-based meshing with threshold on patient 2

The circle of Willis is much more difficult to locate as a result with this technique. With the paint with threshold tool, the work on the geometry consisted on painting slice by slice the pixels that constituted part of the vessels of interest. An effective technique consisted in locating and painting the big vessels which are easier to spot such as the internal carotid arteries and then create the links with smaller arteries. Although it is long to go through all the slices, the cleaning up of the artefact is much easier as the geometry is a clearer picture of the circle of Willis.

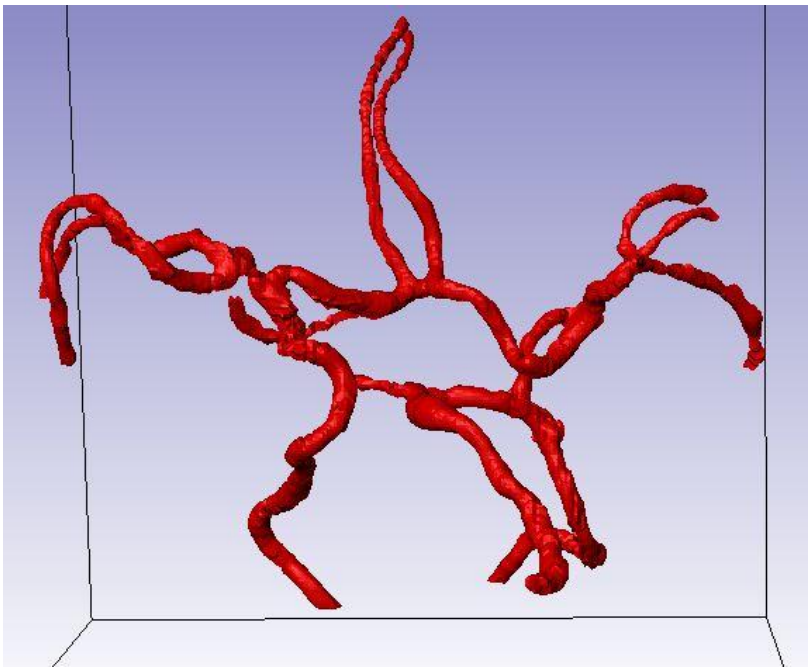

Figure 7 Circle of Willis of patient 1

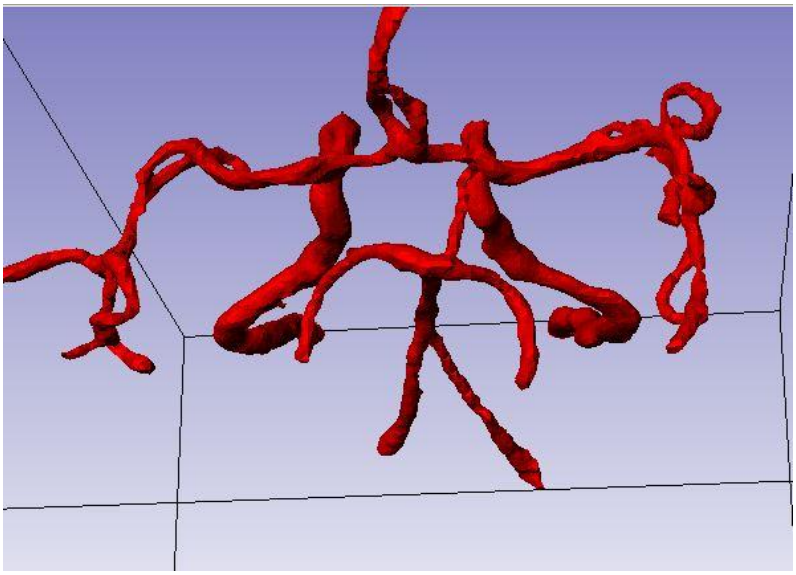

Figure 8 Circle of Willis of patient 2

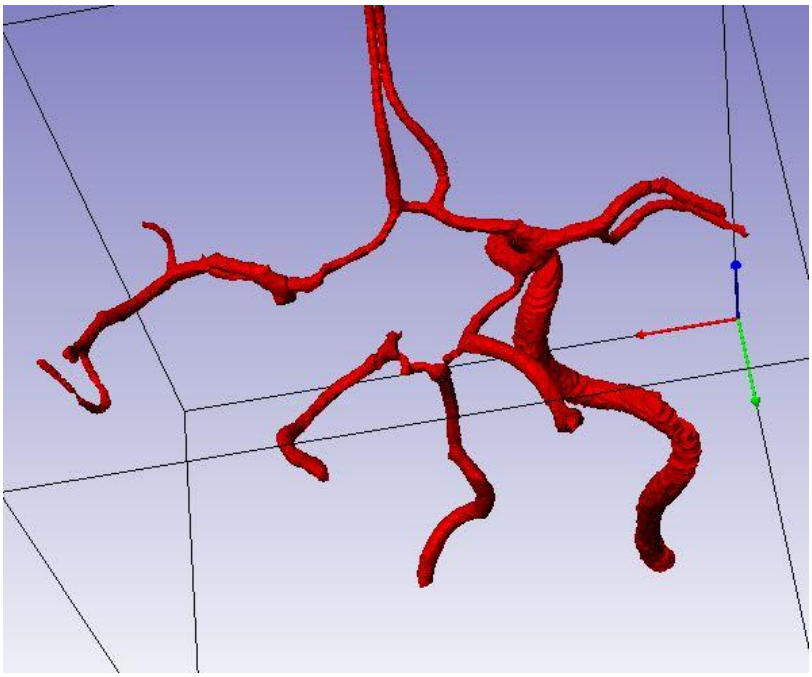

Figure 9 Circle of Willis of patient 3

We can notice on the three pictures that none of the circles of Willis created with ScanIP are complete.

The circle of Willis of the second patient required a little bit more work to obtain a decent reconstruction of the right internal carotid as it was difficult to locate the voxels that pertained to it. According to Dr James, the difficulty to locate this carotid might have been down to the narrowing of the carotid of the patient, a medical condition. The picture on the left shows the incomplete left internal carotid that was created with the paint with threshold tool. To have a more realistic carotid artery, a smoothing filter was applied. As we can see in Figure, the reconstruction is not perfect as the diameter and the shape of the carotid vary abnormally.

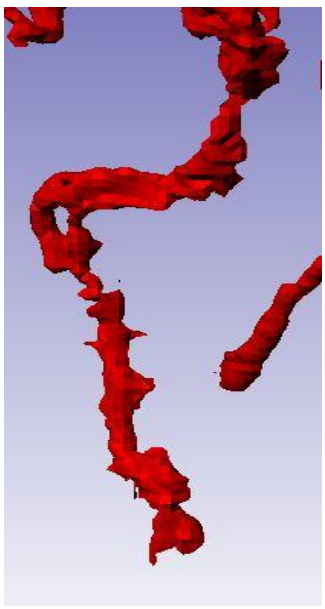

Figure 10 Initial LICA of patient 2

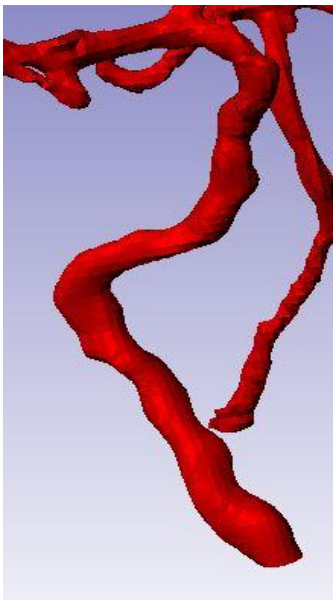

Figure 11 Final and smoothed LICA of patient 2

2.2 Exporting data

One important step after the creation of the mesh is to export it to CFD code so that the mesh is ready to be used in a simulation.

OpenFOAM recognizes two types of surface for the boundary conditions: walls and patches. Therefore all the inlets and outlets in the circle of Willis should be defined as patches. To do so, it was needed to create the surfaces that represent the patches. The technique that was used by Simpleware in a former simulation of the circle of Willis which consisted in intersecting the surfaces that will be considered as inlets and outlets with spheres and cubes, was applied

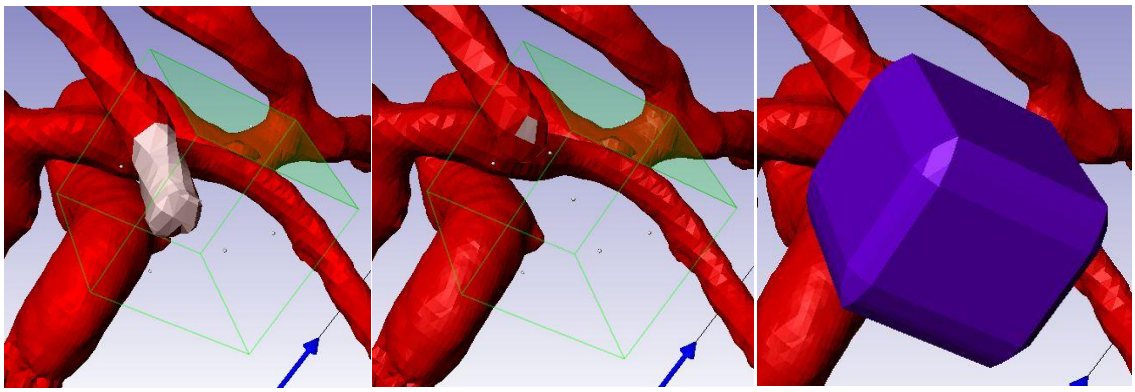

Figure 12 Process to define patches with ScanIP

These spheres and cubes were in fact new masks - sets of voxels - that allowed defining the boundary between them and the blood mask. When testing if ScanIP had assigned the right patches that will be used to define the inlets and outlets of the circle of Willis, it appeared that the wrong surfaces were taken as patches.

This issue resulted from priority the different masks were assigned to for the meshing. It was also necessary to make ScanIP understand that it was to mesh only the part that constituted the blood. The problem was finally solved by applying these two changes in the meshing configuration.

Throughout the internship constant tweaks were applied to the mesh to resolve a few issues that were only noticeable after running simulations. Two problems with the mesh required greater attention.

The first one is the definition of the patches. As said earlier, the patches or boundary conditions were defined as the interface between the blood mask and other masks that were specifically introduce for the purpose of creating the patches. Although it seemed at first glance that the technique described earlier to create the patches was efficient it was not so as simulations showed.

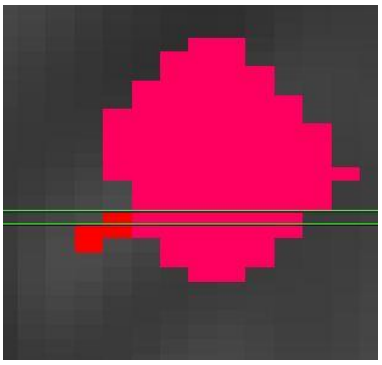

Figure 13 Weak interface with one pixel of each mask touching the other

For the simulation done with constant pressure outlets (picture unavailable) we observed a strange phenomenon. The pressure at all the outlets was set to a same value but we could see that the colour map was not the same for all of them. By looking further into the boundary and p files, it was found that some patches were not taken into account even though they were configured in ScanIP. It turned out that the issue revolved around the “strength” of the interface defining the patch. Although the technique used to define the patches seemed to represent the interface that the outlets are, a look at the slices of the MRI scans showed that the masks did not connect as well as expected. The issue was simply solved by painting the missing voxels of the interface in the different slices.

Another issue related to the strength of the interface between the blood mask and the other masks was the overestimation of the velocity at the outlets. Since the creation of the patches is based on the connected voxels of the different masks, the surface representing the patch was reduced and they were at the tip of a converging vessel. All this led to a dramatic increase of the velocity at the outlets.

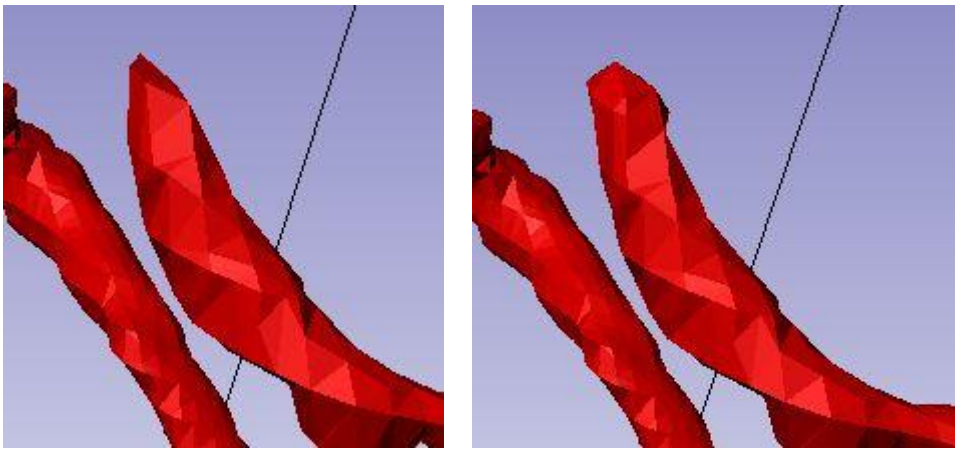

Figure 14 Less converging outlet obtained from painting only one more pixel at the interface

3 CFD study

Computational fluid dynamics in the circle of Willis is a very challenging physical problem. Although computational methods have tremendously improved in the last few years, there are still uncertainties that need to be overcome in order to have results that comply with the reality. There is some parameters' influence that needs to be assessed via sensitivity studies. The boundary conditions and the physical hypothesis occupy a prominent place in the validation of a CFD model.

3.1 Methodology

The boundary conditions occupy a important place in the modelling of hemodynamics in truncated arterial trees. They are supposed to represent the rest of the vascular bed. As we know, the thousands of blood vessels downstream resist the passage of the blood as it flows. Some of the blood is reflected at the bifurcation and the vessels dilate and contract when blood flows. All these phenomena have to be taken into account, at least partly when setting up the boundary conditions on a truncated arterial tree. That is why a substantial amount of time leaned on finding appropriate boundary conditions and implementing them in OpenFOAM when possible.

3.1.1 Modelling of blood

One important task in the study of blood dynamics is the modelling of blood. Blood is known as a non-Newtonian blood. Physically blood is a constituted of plasma, a liquid of physical properties akin to water, and of various particles (red cells, white cells, ...). It is those particles that confer the blood its non-Newtonian properties [1,2,4].

Given the non-Newtonian nature of blood, a viscosity model is needed. However, many studies neglected this non-Newtonian nature [21]. The importance of the Newtonian is particularly highlighted when doing structural analysis of the blood vessel as yield forces are present. The blood was therefore modelled as a Herschel-Bulkley in this study to take this phenomenon into account just like in [12].

$$\mu = k\dot{\gamma}^{n-1} + \frac{\tau_0}{\dot{\gamma}}$$

with k = 8.9721×10^{-3} N.s^n/m^2, n = 0.8601, and τ_0 = 0.0175 N/m². μ represents the viscosity, τ_0 represents the yield stress, and $\dot{\gamma}$represents strain rate. A value of 3.48×10^{-6} Pa.s was set as the initial viscosity.

Just like in many studies [12,13] of blood flow simulations, the blood was modelled as an incompressible fluid with a density of 1050kg/m^3.

Also the selected turbulence model so to speak was laminar. Turbulence in the arteries is indeed a rare occurrence though it can be localized in some of the bigger arteries [4].

3.1.2 Mesh refinement study

Before starting transient flow simulations, mesh sensibility studies were carried out to account for the relevance of the mesh density with regards to the results. The mesh sensitivity studies were using specific settings. The flow was considered steady with a velocity of 0.25m.s^{-1} at the inlets. A constant pressure of 0 Pa was prescribed at the outlets. Since the wall shear stress is an ever important criterion in the development of aneurysm although its clear role has yet to be entirely figured out, the average wall shear stress on walls was used as a parameter to measure the results independence from the mesh. We can see in Figures 15, 16, 17 that for the three patients the average wall shear stress on the walls is an increasing function of the number of cells. The studies could not go further than about 1.1 million cells because of the limitation of the computer (personal computer) that was used to generate the mesh. Figures 15, 16, 17 clearly indicate that mesh independence was not reached. The changes in the geometry when generating the meshes also have a role to play in this mesh dependence but its impact on the result is hard to measure.

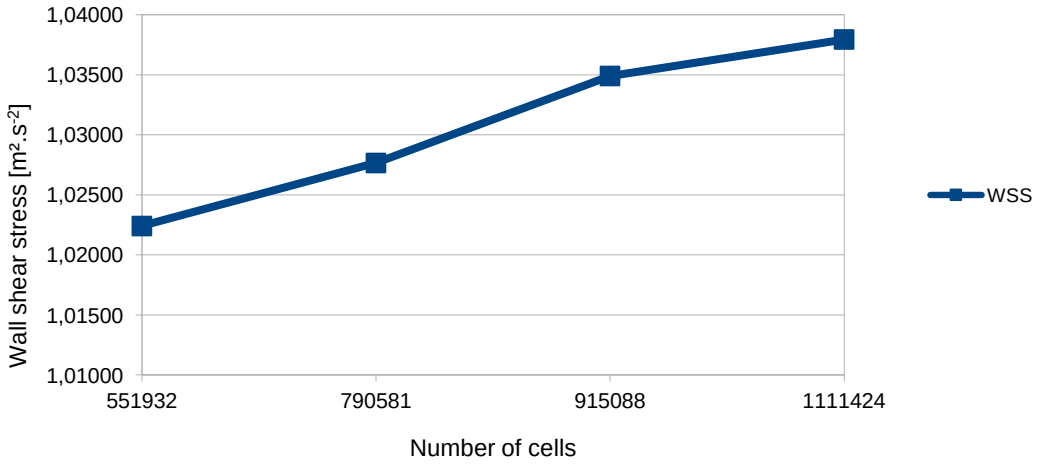

Figure 15 Growing wall shear stress with denser mesh for patient 1

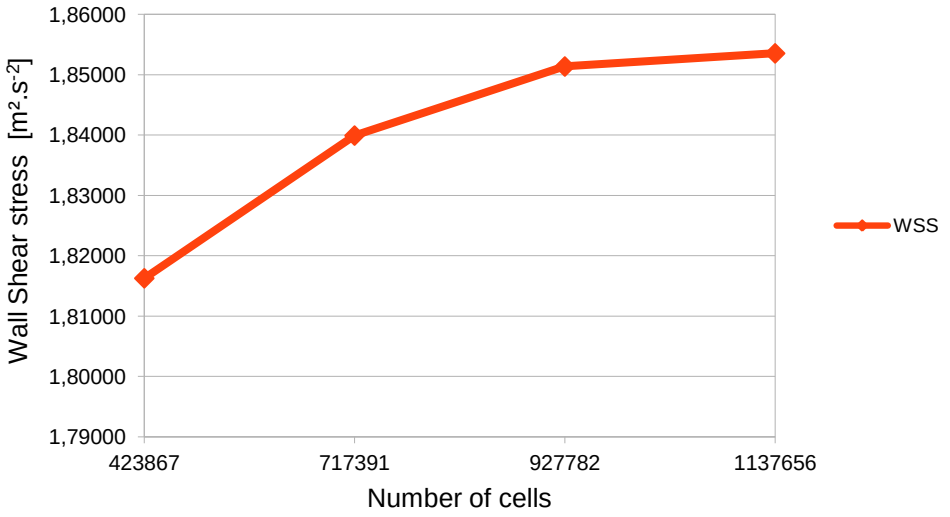

Figure 16 Growing wall shear stress with denser mesh for patient 2

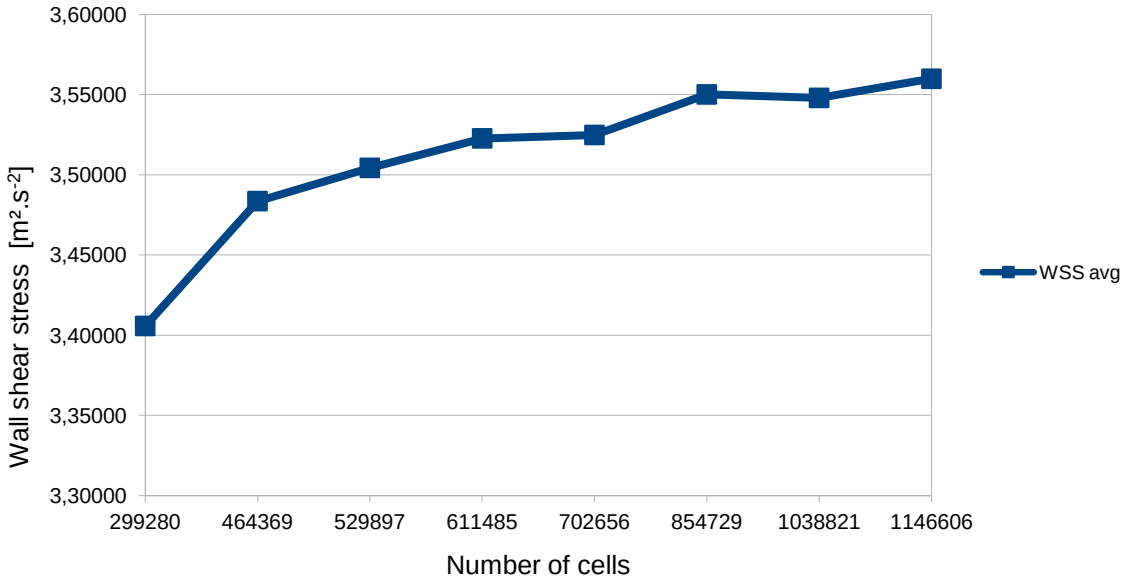

Figure 17 Growing wall shear stress with denser mesh for patient 3

3.1.3 Transient simulations

All the transient simulations were done with the pimpleFoam solver of OpenFOAM.

3.1.4 Mesh density

Although, mesh independence was not reached, various simulations were conducted with the aim of implementing and comparing different boundary conditions. The first transient simulations used meshes of around 400,000 cells. Given the low time steps that were used by the pimplefoam solver, of the order of 10^{-9} to 10^{-7}, it was decided to switch to much coarser meshes to get results in a reasonable time.

Even with coarser meshes, adjustments were needed to have decent time steps. One of the main causes of low time steps was the number of severely non-orthogonal cells. The number of non-orthogonal correctors and the maximum Courant number were increased while keeping the mean Courant number well below 1 to obtain a faster convergence. As a result of these modifications the time steps were mostly contained within the range of values between 10^{-5} and 10^{-4}.

3.1.5 Inlet

For the simulations the inlets used were prescribed velocities based on the Fourier decomposition of the pulse found in [2]. The original signal is represented in Figure 18. The values of the different Fourier coefficient were adjusted in order to obtain a maximum of 0.3 $m.s^{-1}$.

$$f: t \to a_0 + a_1 \cos(\omega t + \varphi_1) + a_2 \cos(\omega t + \varphi_2) + a_3 \cos(\omega t + \varphi_3) + a_4 \cos(\omega t + \varphi_4)$$

Table 1 Fourier coefficients for the inlets

	a_i	φ_i [rad]
$i = 0$	0.33	-
$i = 1$	1	-1.01
$i = 2$	0.97	-2.64
$i = 3$	0.47	2.16
$i = 4$	0.14	1.5

ω is the pulsation defined by $\omega = \frac{2\pi}{T}$. In our case T is the period of the cardiac cycle which was set to 0.92s.

Although the base code of OpenFOAM already offers numerous types of boundary conditions, it is not provided with one that allowed setting up easily the velocity obtained from the Fourier decomposition.

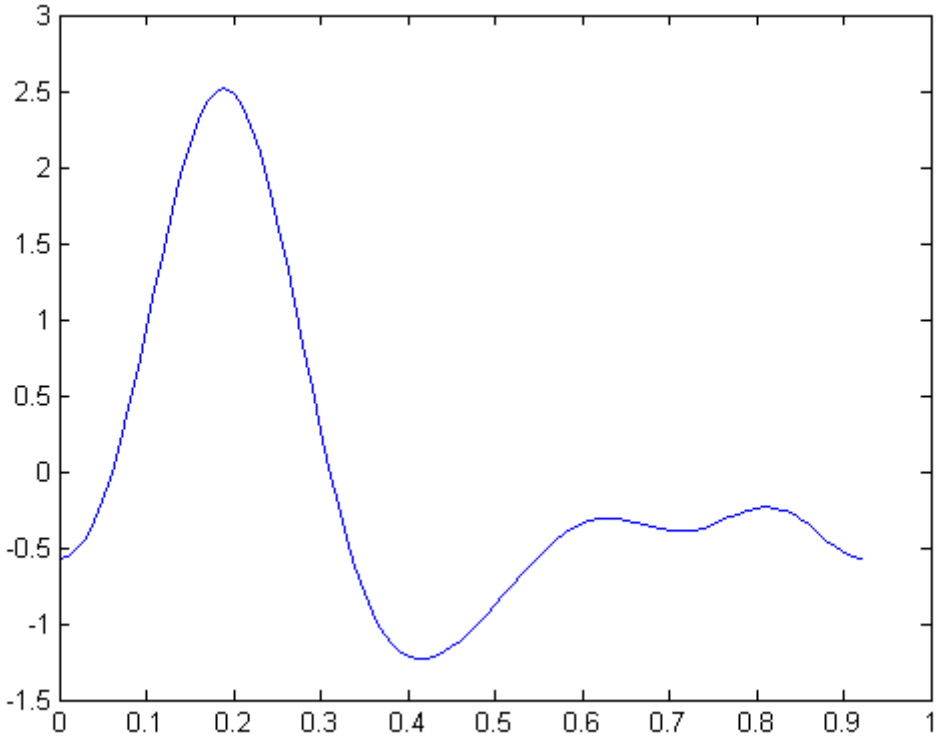

Figure 18 Inlet model function

In order to counter this problem one possible way is to design custom boundary conditions. This was what was first attempted for the inlets boundary conditions. The method consisted in using an existing boundary condition that is similar in some ways to the custom boundary conditions to be created, do the necessary modifications of the code and then compile it to use it with OpenFOAM like any other boundary condition.

The idea of groovyBC was forwarded and the coding of the custom boundary conditions was as a result abandoned. groovyBC is part of the package swak4foam that allows creating boundary conditions based on functions. It made prescribing the velocity at the inlets all the more easier.

In others simulations, the values of the velocity function at the inlet was adjusted so to have volumetric values compatible with the one prescribed in [13]. These adjustments made for a more physiological incoming flow in the circle of Willis.

3.1.6 Outlets

Two types of outlets were tested: the prescribed fixed pressure outlets and the three-element windkessel boundary condition.

The three-element windkessel boundary condition models the outlet with the following circuit:

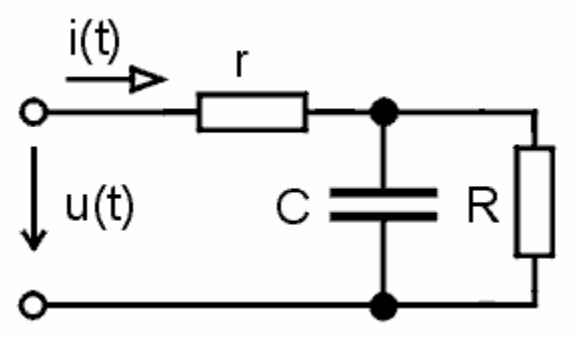

Figure 19 three-element windkessel circuit

From which results the following equation at the outlet:

$$u(t) + RC\frac{du}{dt} = (r + R)i(t) + rRC\frac{di}{dt}$$

u and i are respectively the pressure difference between the outlet and the pressure in venous system $p_v$and the volumetric flow rate at the outlet, r and R are the peripheral resistances and C is the compliance.

For the simulations, the values chosen for r, R, C and p_v were those found in [5], see Table 2. p_v was set to 666.61 Pa.

Table 2 Values of r, R, and C foe the different outlets

	$r\ [10^9 Pa.s.m^{-3}]$	$R\ [10^9 Pa.s.m^{-3}]$	$C\ [10^{-10} m^3 Pa^{-1}]$
PCA	11.08	11.08	0.62
MCA	5.97	5.97	1.16
ACA	8.48	8.48	0.82

The prescribed fixed pressure values used were 0 Pa and 5000 Pa. The constant pressure boundary condition was easily implemented in OpenFOAM since the fixed-value boundary condition is a native OpenFOAM BC.

It was quite a bit trickier to apply the windkessel boundary condition as it is not among the original OpenFOAM boundary conditions. Therefore, it was necessary to find an alternative to implement it.

3.1.7 Implementing the windkessel BC

To solve the equation at the outlets, a first order implicit discretisation of the time derivative of the pressure and the flow rate was used.

3.1.7.1 Implementing with groovyBC

3.1.7.1.1 oldTime function

The first attempts at implementing the windkessel boundary condition involved the groovyBC library.

groovyBC possesses a function that allows to have access to the values of the field the previous time step thanks to the oldTime function. This feature of the groovyBC library was tested on the first unhealthy patient. The principle of the test was to use three outlets A, B and C. At outlet A was prescribed a certain pressure that varied accordingly to time. Pressure at outlet B was set equal to the pressure at outlet A and pressure at outlet C was set to the value of the pressure of outlet A at the previous time step. If the groovyBC boundary condition behaved as expected, the pressure obtained at outlet B should be equal to the pressure at outlet A and the pressure at outlet C should have been the pressure at outlet A delayed by a time step.

The results showed similar values for the pressure at outlet A and B. However, values at outlet C were not resulting from a delayed function of the results at outlet A.

3.1.7.1.2 Stored variables

Given the results of the oldTime function, another feature of the groovyBC library was used to compute the pressure at the outlet: the stored variables. The principle to calculate the pressure from the windkessel boundary conditions is the same as the one that was to be used with the oldTime function which was using values of pressure and flow rate at the previous time step to calculate the time derivatives. The values of flow rate and pressure at the previous time step were saved in the two stored variables that were precisely created for this purpose.

A simple case was set up to test the behaviour of the stored values. The case consisted of a fluid flowing through a square-based cylinder. The geometry is shown in Figure 20.

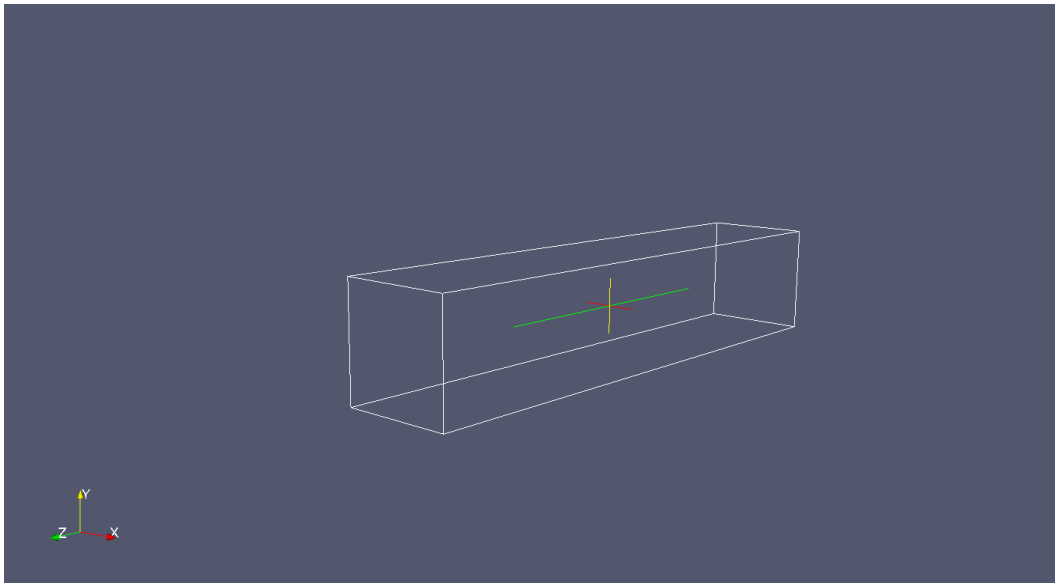

Figure 20 Square-based cylinder geometry for the windkessel test-cases

The boundary condition imposed at the inlet was a velocity represented by a linear function of time. The outlet was a windkessel boundary condition for which the value of p_v was set to 0. r, R and C were all set to 1. Given the relative simplicity of the case, theoretical values of the pressure were calculated for the boundary. The pressure obtained from the

discretized equation was also calculated using a spreadsheet. This allowed comparing and judging the effectiveness of the stored variables of the groovyBC library.

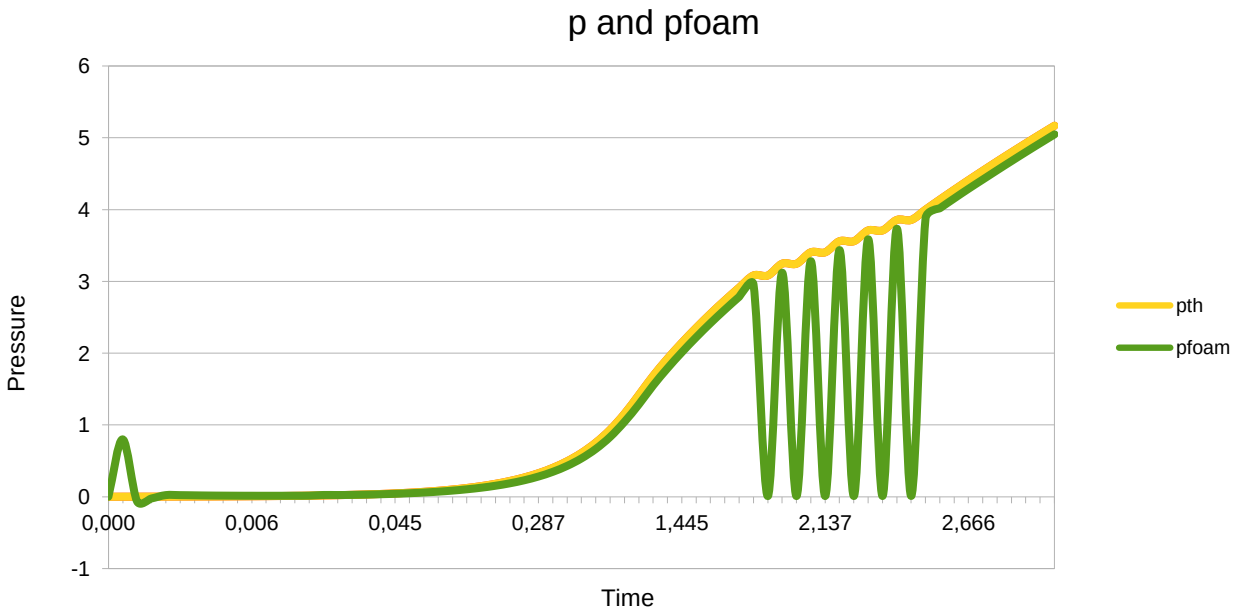

Figure 21 Average pressure calculated with OpenFOAM and spreadsheet at the outlet showing inadequate OpenFOAM-calculated results

The spreadsheet's pressure curve, pth, and the curve obtained from OpenFOAM calculation are quite similar but we can notice six peculiar points in the pressure calculated by OpenFOAM that do not follow the general trend of the theoretical curve. By having a closer look at the result, it was noticed that the stored value of pressure was unexpectedly set to 0.

Note that the simulation was done with an adjustable time step and the discretised "solution" in no way represents the theoretical solution.

To investigate further into this issue, other cases were set up with differing velocity inlet functions. The three new functions introduced as inlet velocity functions were : t->t^2, t->exp(t) and t->sin(t).

The results of the discretised equations calculated with OpenFOAM and the spreadsheet were plotted to analyse the behaviour of the stored variables. The theoretical values of the pressure were adjusted by replacing the time step with the write interval. When plotting the curve of the pressure calculated by the OpenFOAM solver and the curve of the adjusted theoretical pressure, a very good likeness was observed for the cases with exponential and parabolic velocity inlet.

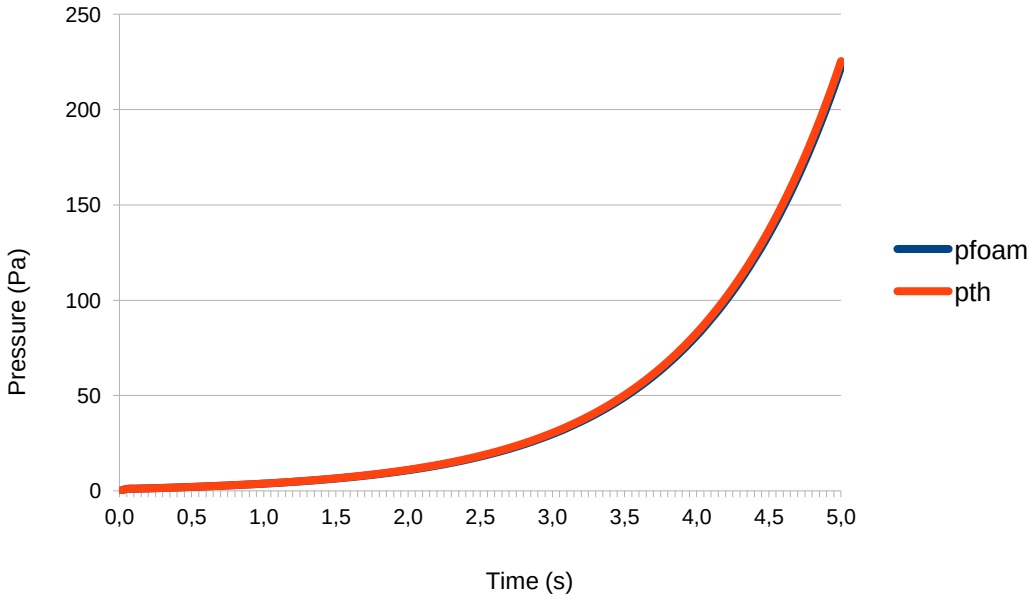

Figure 22 Average pressure calculated with OpenFOAM, in blue, and spreadsheet, in red, at the outlet with exponential inlet

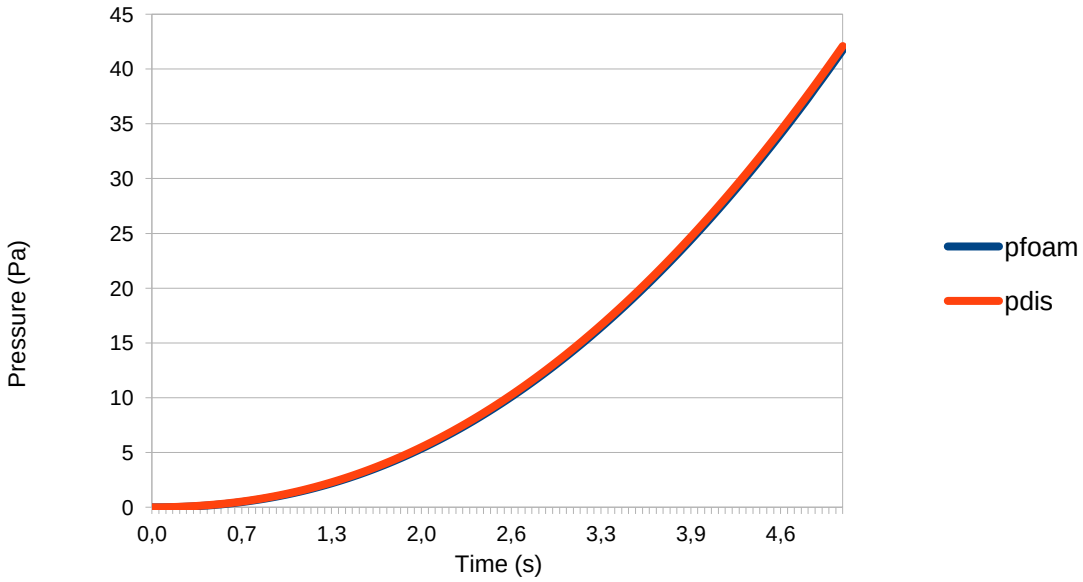

Figure 23 Average pressure calculated with OpenFOAM, in blue, and spreadsheet, in red, at the outlet with parabolic inlet

For the sine velocity inlet, the comparison between calculations and theoretical results was not as good.

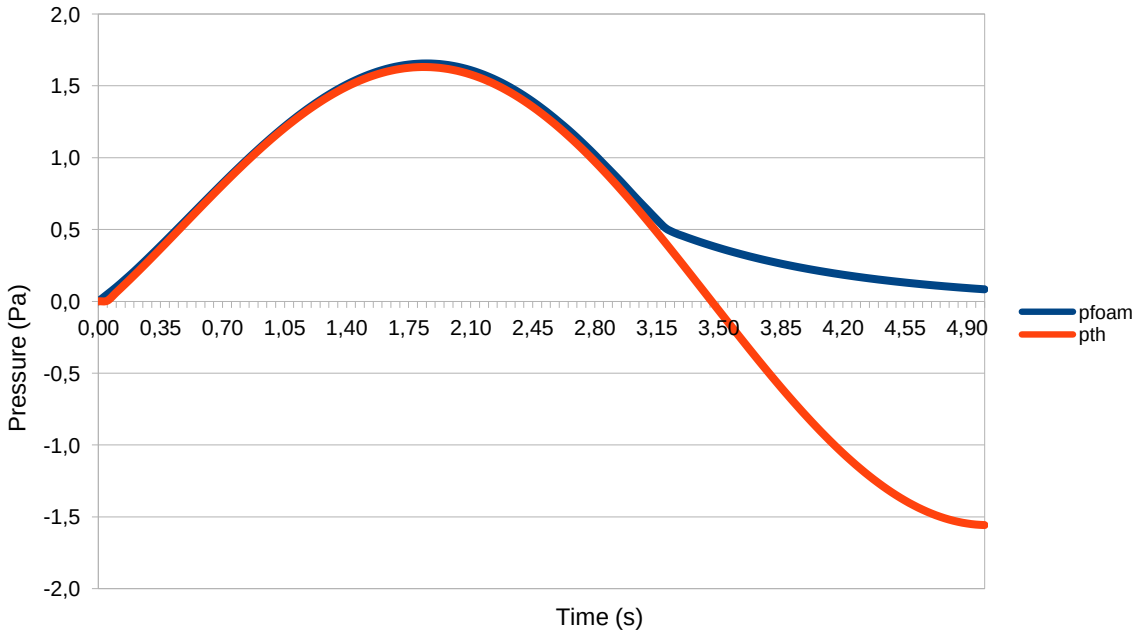

Figure 24 Average pressure calculated with OpenFOAM, in blue, and spreadsheet, in red, at the outlet with sine inlet

The results are quite alike at the start but start to diverge from each other from the time t=3.2s onwards. Again, by looking at the files created during the calculations we see that the stored variable supposed to store the value of the flow rate is set to 0 for most of the times after t=3.2s.

Given the unexpected and erratic behaviour of the stored variables of groovyBC, although calculations were done using them, their representation of the windkessel boundary condition was deemed not reliable enough to extract any conclusive result from the calculations.

3.1.7.2 Hard-coded windkessel BC

A third way was explored to implement the windkessel boundary condition in OpenFOAM. Since OpenFOAM is an OpenSource toolkit, modifying it and adding bits to it, like the groovyBC library, is a common practice. Therefore, the third way to implement the windkessel boundary condition was to code and compile it to use it with OpenFOAM. To code the windkessel boundary condition and add it as one of the available boundary conditions with OpenFOAM solvers, basic guideline of [16] were followed. The gist of it was to copy and modify a boundary condition similar in some ways to the one to be implemented.

Among the 31 native boundary conditions of OpenFOAM-2.1.0, the advective boundary condition seemed the closest to the windkessel boundary condition as the advective equation involves time derivative of the variable in question.

At first, both the pressure and the flow rate derivatives were discretised but the results – not reported in this report – were not good enough. Therefore, it was decided to calculate the time derivatives thanks to the differencing schemes already implemented in OpenFOAM, so in a way not discretising directly the time derivatives. To do so, it was necessary to

calculate the derivative in the time loop as calculating the time derivatives directly in the files containing the boundary conditions led to compilation errors. It was needed to modify the pimpleFoam solver by calculating the derivative of the flow rate and the pressure and storing the values in two new variables. The derivative of the velocity gave values in a normal range but the derivative of the pressure exploded leading to floating exception. The problem was solved with discretising directly the pressure derivative.

To test the boundary condition, the cases of the cylinder were used again and the results were compared to the actual solution of the cases. The differential equation of the windkessel boundary condition was solved with each of the three test functions. The solutions are reported in Table 3.

Table 3 Theoretical solutions to test-cases

Inlet function	Solution
$t \to t$	$t \to 2t - 1 + e^{-t}$
$t \to t^2$	$t \to 2(t^2 - t + 2) - 4e^{-t}$
$t \to exp(t)$	$t \to 1.5e^{t} + 0.5e^{-t}$
$t \to sin(t)$	$t \to 0.5(3\sin(t) - \cos(t) + e^{-t})$

Just as observed with the stored variables of the groovyBC library, the pressure calculated with the pimpleFoam solver was in agreement with the theoretical pressure with the polynomial and exponential test functions. It was slightly different for the sine velocity inlet as it was in agreement until a certain point from which the pressure was set to 0.

It was found out that this point was in fact the one from which the velocity was negative. The nullification of the pressure was a consequence of the default outlet boundary condition for the velocity called inletOutlet whose behaviour was similar to a zero gradient boundary condition as long as the velocity was positive. When there is a reversed flow, the boundary condition switches to a fixed value boundary condition. This is what led to diverging values for both the stored values cases and the hard-coded windkessel boundary condition. The result obtained by switching from the inletOutlet boundary condition to a zero gradient one for the velocity of the outlet showed that both the stored values and the hard-coded windkessel boundary condition were good approximations of the theoretical pressure at the outlet with the three different test functions. The hard-coded windkessel boundary condition is just a slightly better approximate.

On the following graphs, were plotted the solutions to the windkessel equation the stored, hard-coded and direct calculation methods with the correct boundary conditions.

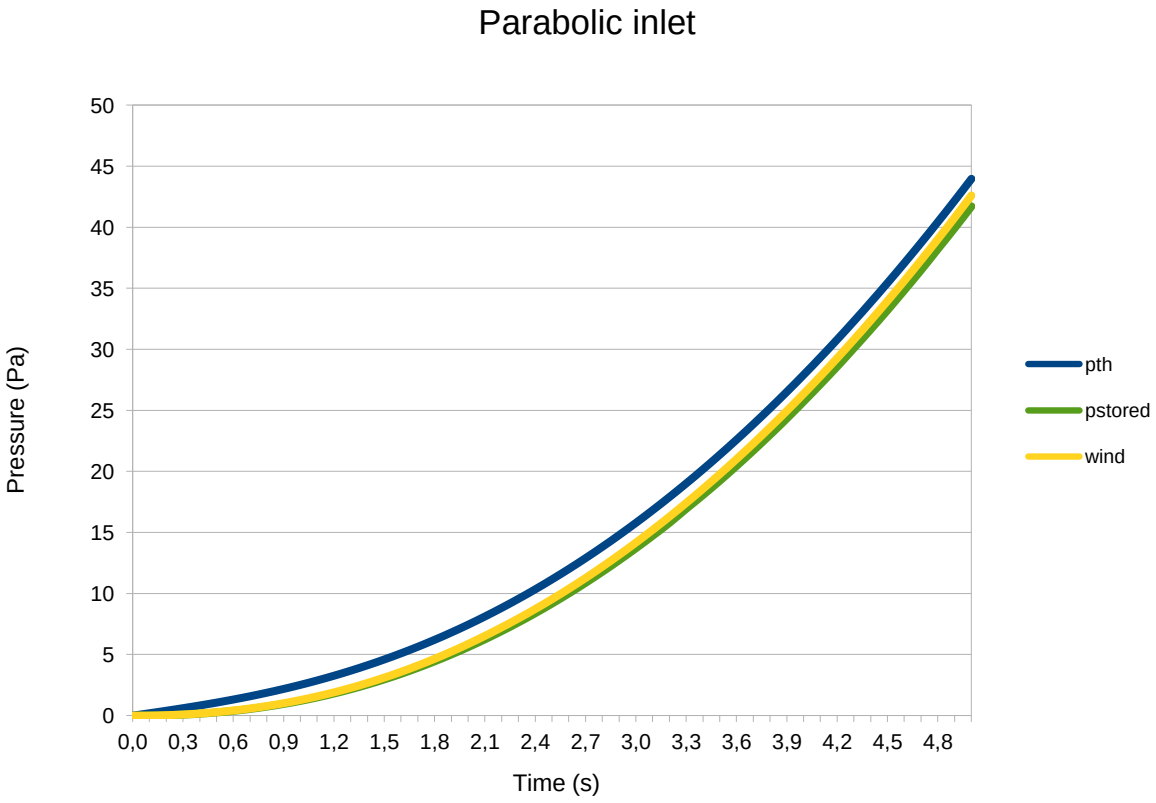

Figure 25 Theoretical pressure in blue, hard-coded windkessel pressure in green and groovyBC windkessel pressure in yellow with parabolic inlet

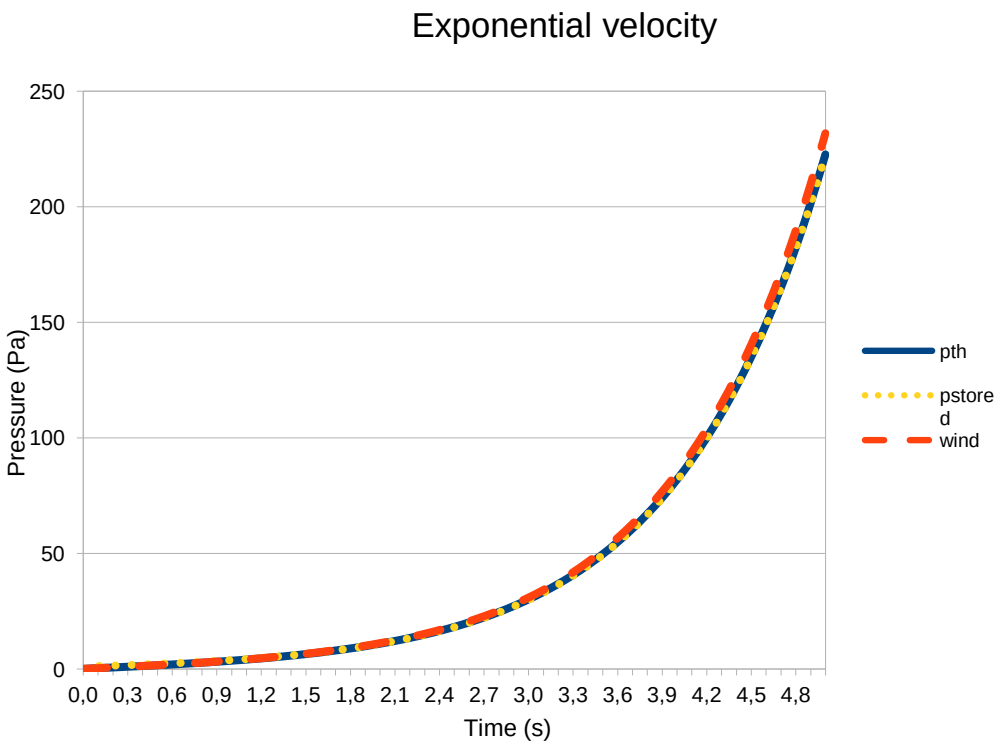

Figure 26 Theoretical pressure in blue, hard-coded windkessel pressure in red and groovyBC windkessel pressure in yellow with exponential inlet

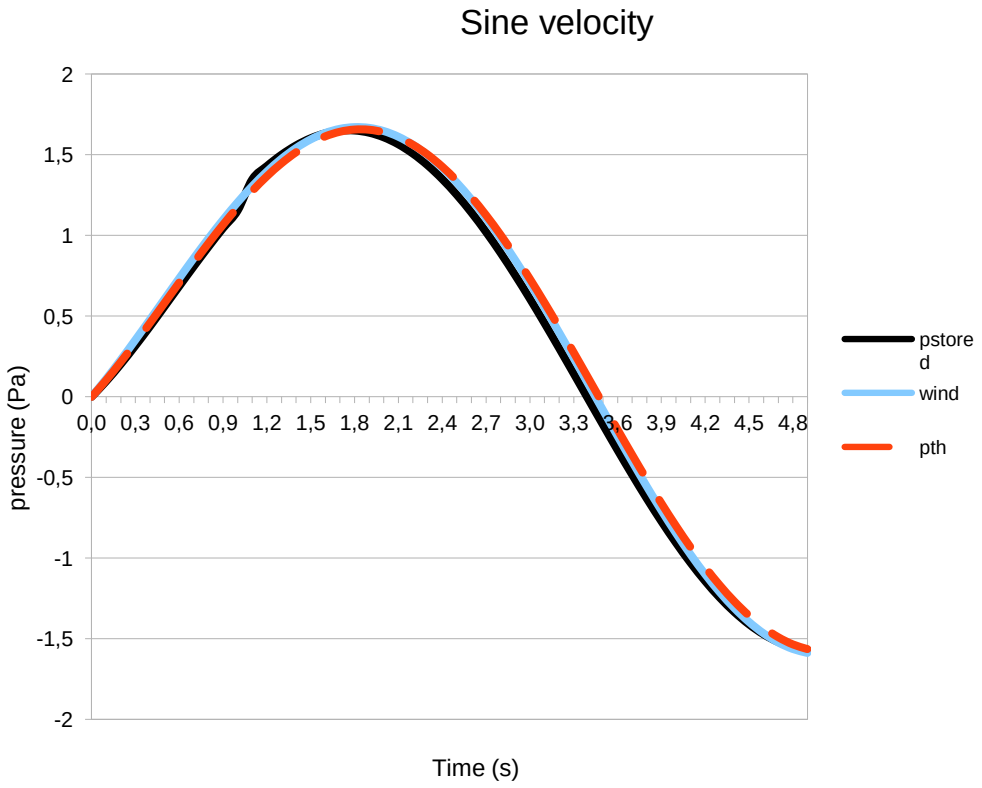

Figure 27 Theoretical pressure in red, hard-coded windkessel pressure in blue and groovyBC windkessel pressure in black with sine inlet

3.2 Analysis

3.2.1 Stability and convergence

To analyse if the calculation had converged and if they were giving stable results two criteria can give us hints. First for the stability, we have the Courant number which is defined in equation below. Generally the Courant number has to be kept at low values in order to assure the stability of the calculations. Adjustments on the maximum Courant number were needed to obtain stable calculations. In a few test simulations the Courant number exploded because of the number of severely non-orthogonal cells and high maximum Courant number.

$$C = \frac{u_x \Delta t}{\Delta x} + \frac{u_y \Delta t}{\Delta y} + \frac{u_z \Delta t}{\Delta z}$$

To control the convergence of the results a couple of key characteristics (wall shear stress, pressure and flow rate at the outlets, pressure on walls ...) were plotted to see if we obtained periodic results. The results of simulation that contained an error in the value of viscosity showed a good periodicity which led to believe that the cases were well set up to obtain converging results. Unfortunately, for the simulations with correct values, only a handful of time steps were written by the end of the project, so results are quite limited for analysis.

3.2.2 Results and discussion

It is difficult to draw any conclusion on the validation of the simulations given the lack of data on the patients (flow rates and pressures) but, with limited data, we tried to verify if some of the generally known behaviours of blood flow and pressure were reproduced by analysing the velocity, the pressure and the wall shear stress.

3.2.2.1 Pressure

The pressure distribution of the simulations with prescribed pressure and windkessel outlets were plotted for comparison's sake.

We can see in Figure 28 and Figure 29 that the pressure with the windkessel boundary condition is much more homogeneous than the pressure with the prescribed pressure outlets for the third patient but it is only due to the scale that was automatically adjusted to the range of value. If we modify the scale we get pretty similar pictures for the two boundary conditions. What we can tell from Figure 28 and Figure 29 is that the range of pressure is broader for the windkessel boundary condition. It is assumed that the low pressure points are a few points localised at the outlets as the points presenting a low pressure were not distinguishable when the scale was changed to lower values.

All in all the pressure distributions are pretty similar for the two boundary conditions. No trend can be extrapolated from the results presented in Table 4. However, we can notice that for the windkessel boundary condition, we have higher pressure in the RMCAs than the LMCAs. This pressure difference may result at later times for different flow patterns depending on the boundary condition used.

We can see on the picture of the prescribed pressure boundary condition that we have a greater pressure on the right side. Given that the left internal carotid is missing, it is understandable that we have less pressure on the left side of the circle of Willis.

For the second patient, the pressure is also more homogeneous with the windkessel boundary condition than with the prescribed pressure due to the broader range of value. However it can be noted that the pressure with prescribed pressure in patient 2 is better distributed than the pressure with prescribed pressure with patient 3. It mainly results from the fact that the circle of Willis of the second patient is more complete and a missing inlet such as an internal carotid, which is one the main supplier of blood, has a big impact on the pressure distribution.

For patient 2 we can also notice different zones of low pressure. In the zones of low pressure of Figure 30, we actually have a negative pressure resulting from a reversed flow. This is a massive difference between the two types of boundary conditions applied.

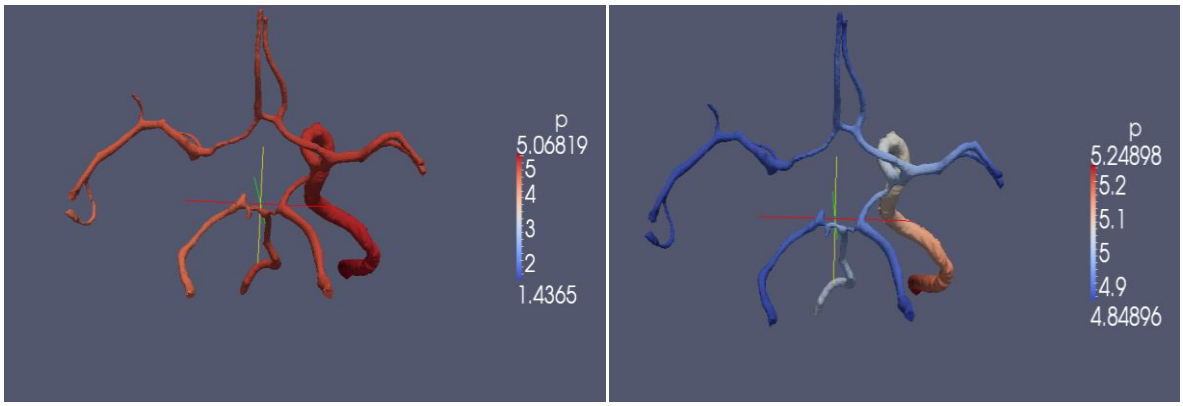

Figure 28 Pressure for patient 3 at t=0.05s, windkessel BC on the left and constant pressure boundary condition on the right

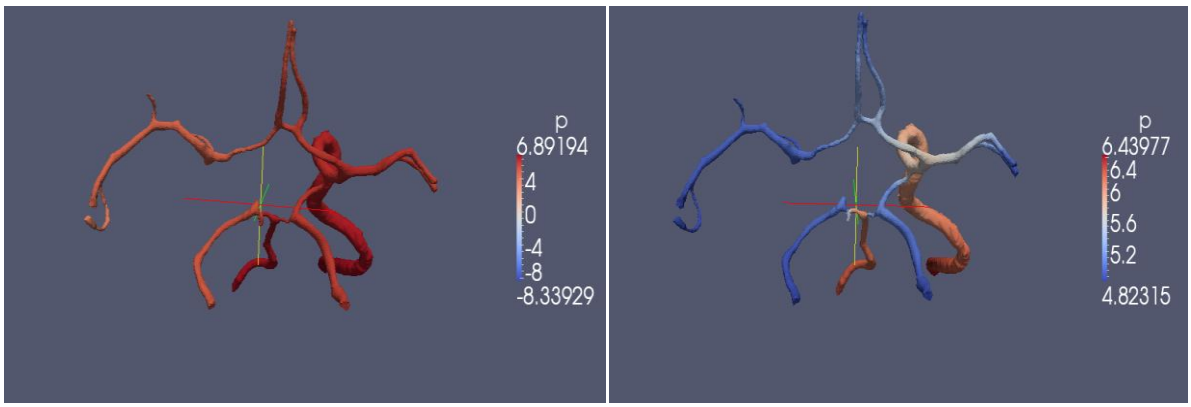

Figure 29 Pressure for patient 3 at t=0.1s, windkessel BC on the left and constant pressure boundary condition on the right

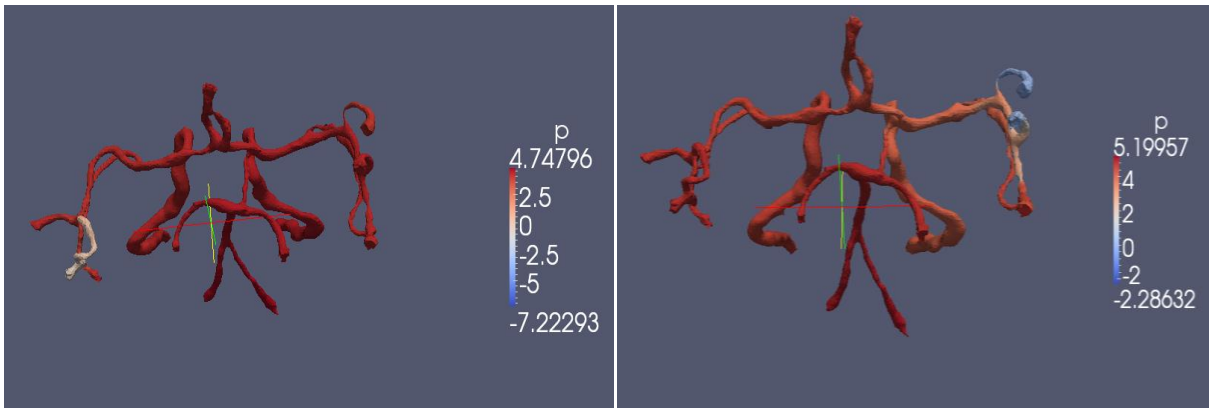

Figure 30 Pressure for patient 2 at t=0.05s, windkessel BC on the left and constant pressure boundary condition on the right

Table 4 Pressure at the outlets of patient 3 with windkessel and constant-pressure BCs

Artery	Pressure at t=0.05s [m^2/s^{-2}]	Pressure windkessel at t=0.05s [m^2/s^{-2}]	Pressure at t=0.1s [m^2/s^{-2}]	Pressure windkessel at t=0.1s [m^2/s^{-2}]
ICA	5.24749	5.06667	6.41996	6.87215
BA	5.02189	4.7493	6.23647	6.68867
RMCA1	4.85	4.68216	4.85	5.37633
RMCA2	4.85	4.68473	4.85	5.66655
LMCA1	4.85	4.60752	4.85	4.42511
LMCA2	4.85	4.39652	4.85	3.44173
LPCA	4.85	4.43207	4.85	4.88224
RACA	4.85	4.58772	4.85	4.62451
LACA	4.85	4.60018	4.85	4.60168

Table 5 Average pressure on the wall for patient 3 with windkessel and constant-pressure BCs

	Pressure on wall at t=0.05s [m^2/s^{-2}]	Pressure on wall at t=0.1s [m^2/s^{-2}]
constant pressure	5.00118	5.64051
windkessel	4.77368	5.92961

3.2.2.2 Flow rates

Here are the results obtained in terms of flow rates at the outlets for patient 3

Table 6 Flow rates at the outlets of patient 3 with windkessel and constant-pressure BCs

Artery	Flow rate at t=0.05s [m^3/s]	Flow rate windkessel at t=0.05s [m^3/s]	Flow rate at t=0.1s [m^3/s]	Flow rate windkessel at t=0.1s [m^3/s]
RMCA1	1.18315e-08	8.32132e-09	5.15324e-07	4.61814e-07
RMCA2	4.91683e-08	5.03493e-08	8.01325e-07	5.34732e-07
LMCA1	6.31613e-09	-2.67912e-07	5.57339e-08	-6.02675e-07
LMCA2	1.2711e-09	2.66525e-07	1.46303e-08	7.58867e-07

LPCA	8.70279e-09	3.2224e-08	1.23744e-07	1.59594e-07
RACA	1.2564e-08	2.8081e-09	2.16887e-07	3.12253e-07
LACA	1.12558e-08	7.23679e-09	1.38502e-07	2.05681e-07

Apart from the RMCAs and one of the LMCAs (LMCA1), the flow rate is superior with the windkessel boundary condition. For the LMCA in question we have an odd negative value. By plotting the velocity vectors around LMCA1, we can see a little bit of reversed flow but what stands out even more is the way the blood seems to flow in all kind of directions from LMCA1. The little asperity on the patch representing LMCA1 might explain this odd distribution of velocity but we can see in the case with the prescribed pressure that blood seems to flow normally.

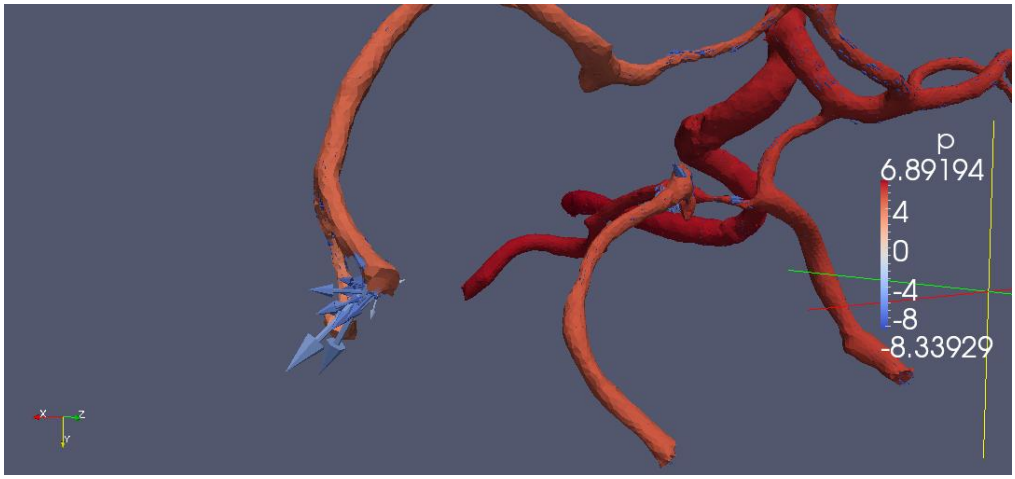

Figure 31 Velocity vectors at LMCA2 for patient 3 with windkessel BC at t=0.05s

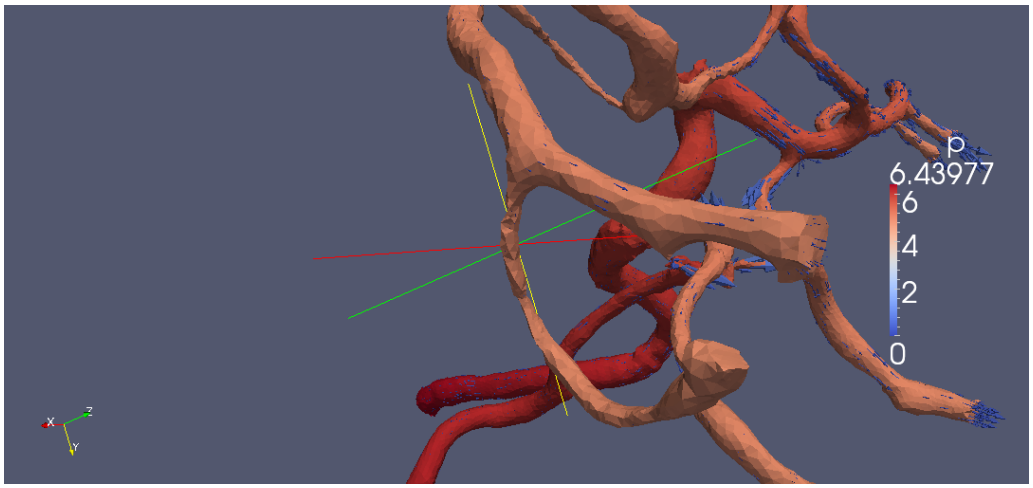

Figure 32 Velocity vectors at LMCA2 for patient 3 with constant-pressure BC at t=0.05s

For the third patient as shown by the velocity vectors and the streamlines, the left side of the circle of Willis is underfed in blood which is a consequence of the missing ICA.

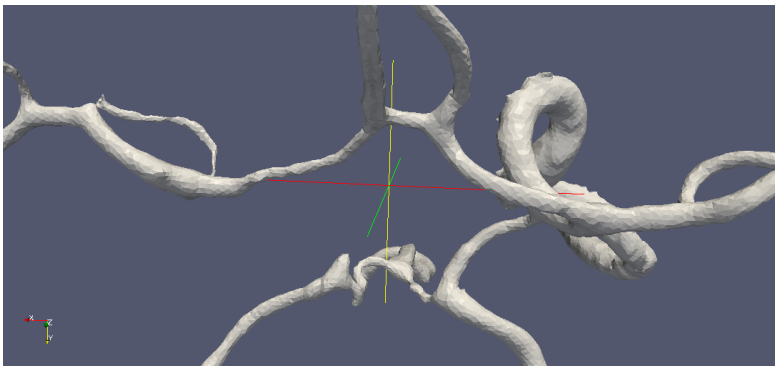

Figure 33 Circle of Willis of patient 3

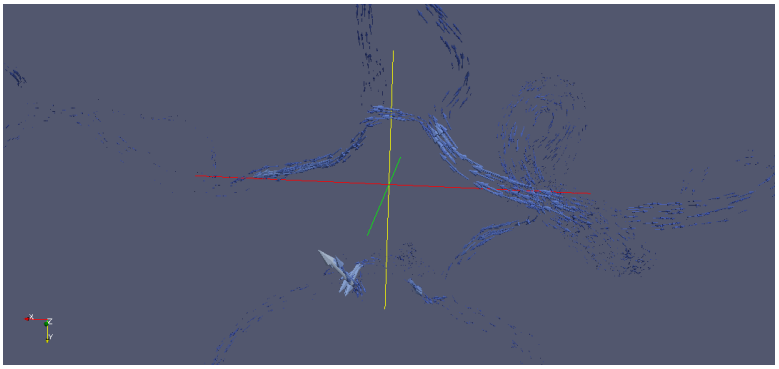

Figure 34 Velocity vectors around the MCA of patient 3 at t=0.1s

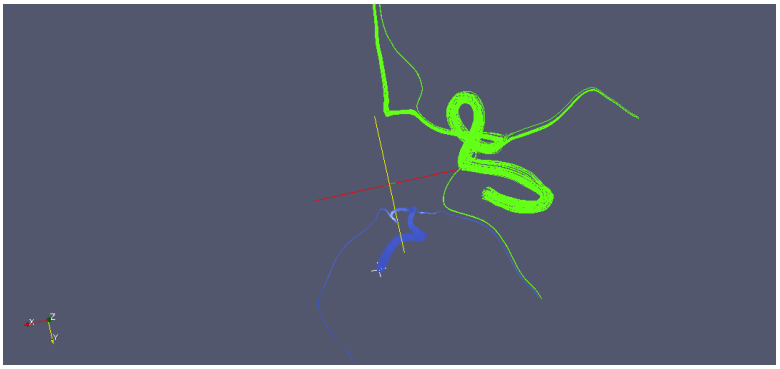

Figure 35 Streamlines of patient 3 at t=0.1s

3.2.2.3 Wall shear stress

For the two images of Figures 36 and 37, the range of value of wall shear stress was set between 0 and 30 Pa (between 0 and 0.0286 $m^2.s^{-2}$) as according to [13] a wall shear stress superior to 30 Pa is considered pathological is sufficient to induce aneurysm. Table 7 and Figure 38, 39 show that wall shear stress seems to be greater with windkessel boundary conditions for the third patient. The same conclusion can be drawn with the second patient given that the average wall shears stress at the time t=0.05s is three to four times superior with the windkessel boundary condition.

On patient 3, we can see that the zones of high wall shear stress are the zones of low pressure.

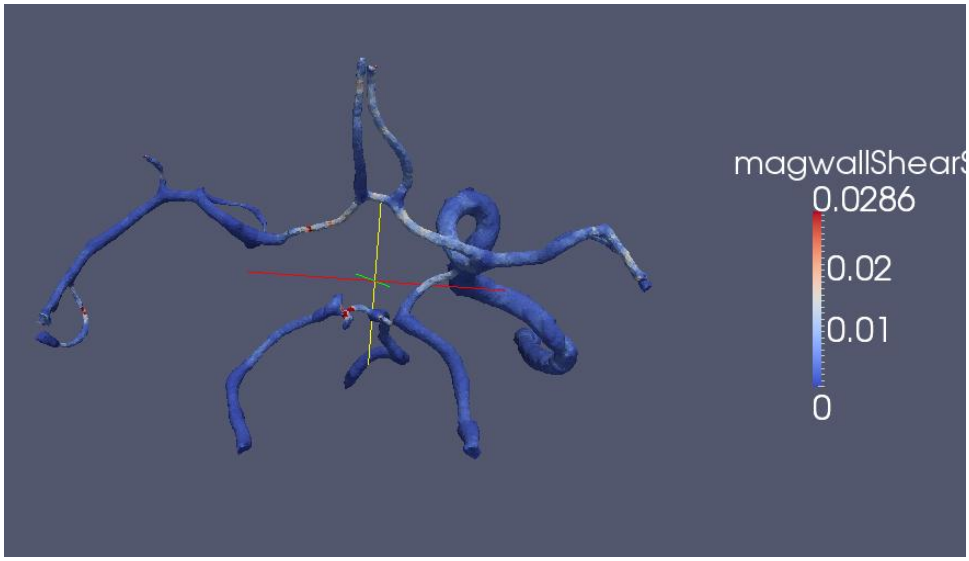

Figure 36 Wall shear stress of patient 3 at t=0.1s with windkessel BC

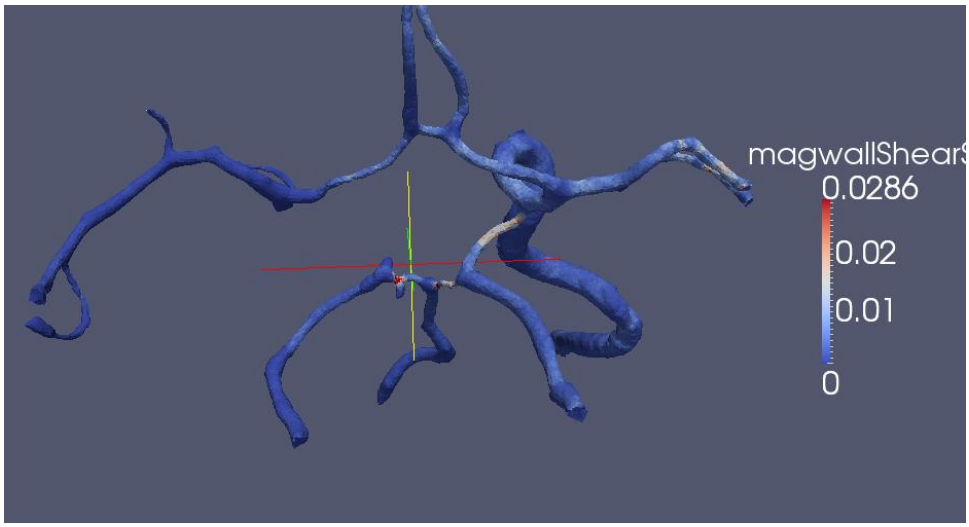

Figure 37 Wall shear stress of patient 3 at t=0.1s with constant-pressure BC

Table 7 Average wall shear stress on walls for patient 3 with windkessel and constant-pressure BCs

	WSS at t=0.05 [m^2/s^{-2}]	WSS at t=0.1 [m^2/s^{-2}]
constant pressure	0.000238825	0.00251343
windkessel	0.000284656	0.00284922

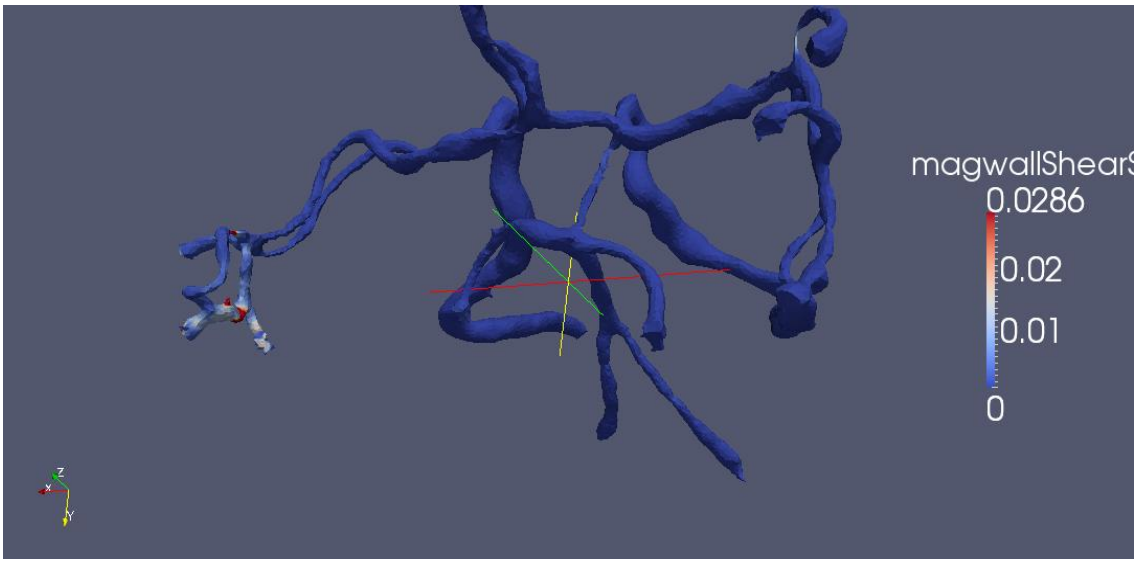

Figure 38 Wall shear stress of patient 2 at t=0.05s with windkessel BC

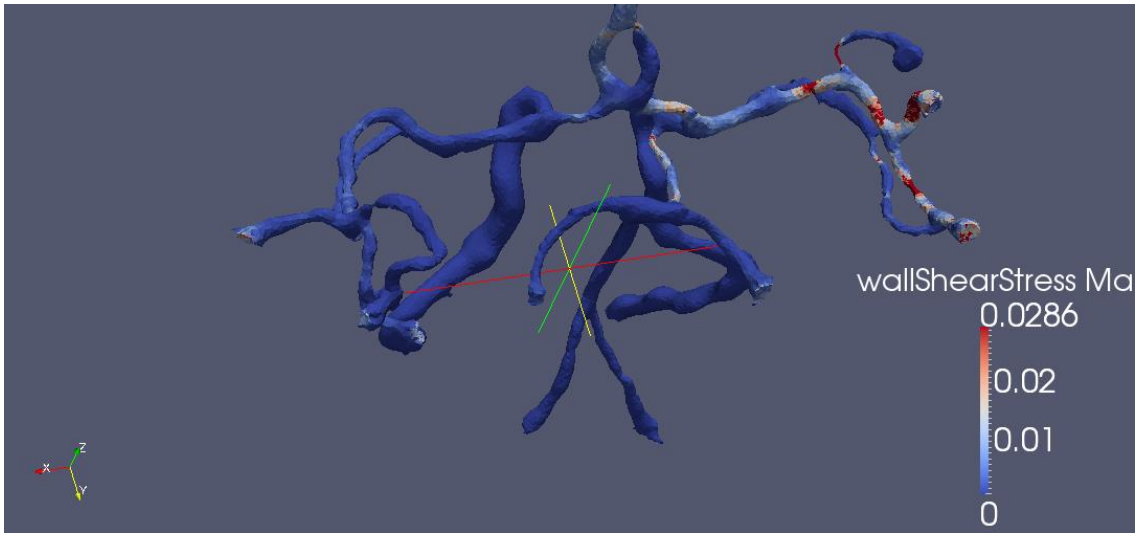

Figure 39 Wall shear stress of patient 2 at t=0.05s with constant-pressure BC

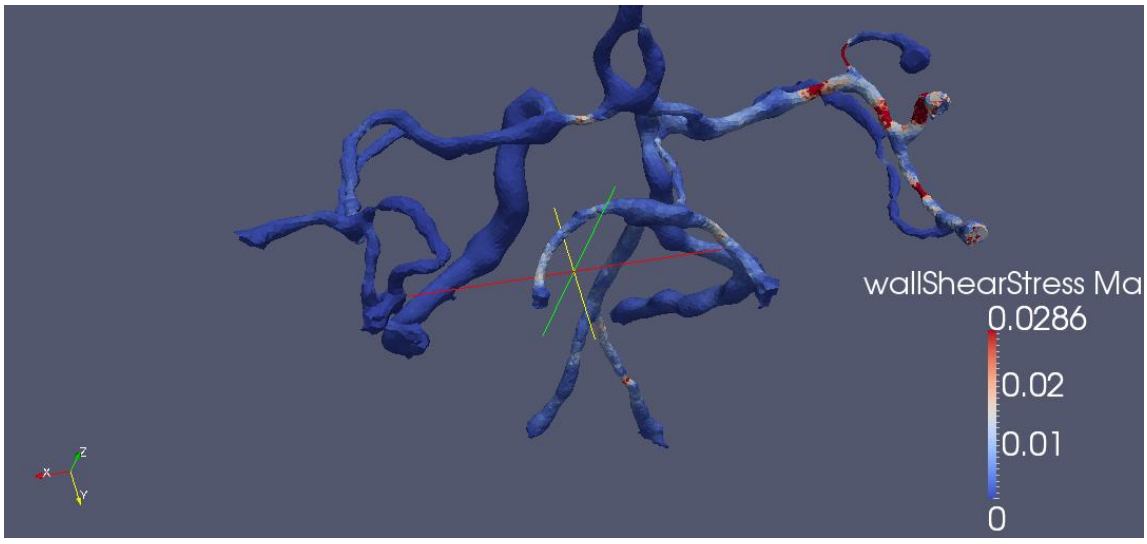

Figure 40Wall shear stress of patient 2 at t=0.15s (peak systole) with constant-pressure BC

Table 8 Average wall shear stress on walls for patient 3 with windkessel and constant-pressure BCs

	WSS at t=0.05 [m^2/s^{-2}]
constant pressure	0.000540377
windkessel	0.00196844

3.2.2.4 Velocity profiles

Regarding the flow, we verified if some of the findings from [11] regarding the velocity profile at the bifurcations were valid in our model. According to [11], the velocity profile at the ICA-MCA junction is generally flattened or presents a local minimum at the centre where the blood is supposed to impinge on the wall. It is supposed that this shape of the velocity profile is induced by the bends in the ICAs and explains why these bifurcations are generally not zones that tend to develop aneurysm. Also according to [11] the MCA bifurcations are zones of high risks given that it is the blood with the highest velocity that impinges on the vessel thus the high risk of aneurysm.

For patient 2, the velocity profiles at the key points were plotted to verify this assertion. The velocity has indeed a flattened profile, represented in Figure 41, at the ICA-MCA junction and it seems that it is the blood at the maximum speed that impinges on the vessel as shown in Figure 42. The slower blood flows normally through the efferent vessels.

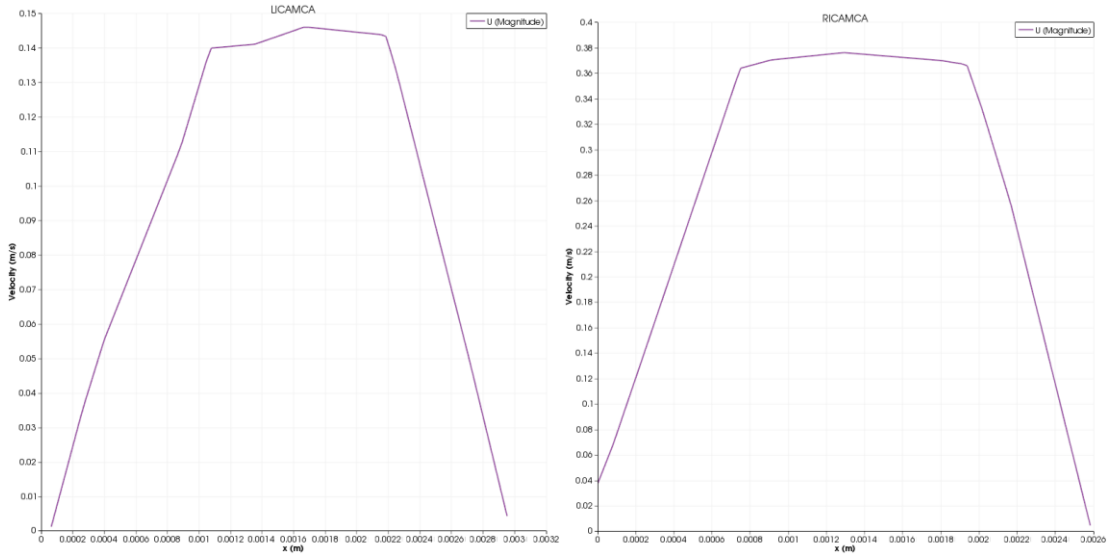

Figure 41 Flattened velocity profiles at the left and at the right ICA-MCA of patient 3 a time t=0.15s (peak systole)

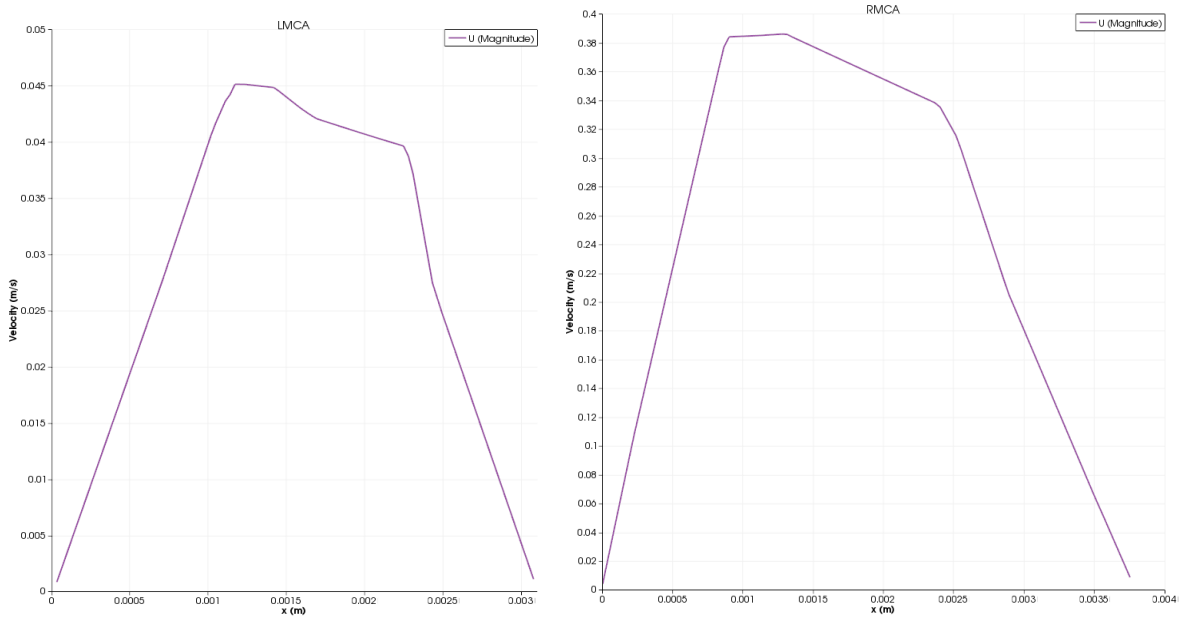

Figure 42 Velocity profiles at the left and at the right MCA bifurcations of patient 3 a time t=0.15s (peak systole)

3.2.3 Suggestions for future work

Without a medical background it is difficult to analyse the data especially when deprived of actual data on pressure and flow rate of the patients. Since the flow rates of the patients were measured at certain particular points of the circle of Willis by the Royal Devon and Exeter Hospital, these data will come in handy to assess the effects of the boundary conditions and validate the CFD models. They were not available during the project.

Although most targets set for this project were reached, there are many things that could be done to improve the model and the speed of the calculations.

From the observations made during this internship, the Courant number, along with the number of severely non-orthogonal faces are the main culprits. It shows that there is some more work to do particularly on the mesh and the case-setting. The limited results tell us that further adjustment should be done on the mesh with the aim of reducing the number of severely non-orthogonal faces. A high number of severely non-orthogonal faces made the simulations crash and to remedy to that the number of correctors was increased leading to greater computational cost. On the case-setting, the right balance should be found so that the time steps do not become too small and the Courant number stays low.

In a way, all the boundary conditions were quite symmetrical even though the shape of the circle of Willis of the unhealthy patients in particular was far from being symmetrical. There are also more physiological boundary conditions than the windkessel one that could be implemented in OpenFOAM. Moreover, other inlet and outlet boundary conditions could be looked at to take into account reflections at the bifurcation.

Once the results have converged it might be worth checking if there are major difference when using coarse meshes and steady calculations to get quicker results.

Conclusion

During this internship, a survey of the different boundary conditions that could be applied to the circle of Willis to make for a more physiological modelling has been done. Based on this survey, the most realistic boundary condition that could be implemented in OpenFOAM without an excessive computational cost was found to be the windkessel one. The windkessel boundary condition was successfully implemented in OpenFOAM using two possible methods: one hard-coded and another with the use of the groovyBC library. Both methods showed good results in the test-cases.

Besides looking at the boundary conditions, this project aimed at also applying them to the circle of Willis of three unhealthy patients. The mesh of the circle of Willis of each of the three patients was created and they are ready to use for simulation with OpenFOAM for future work.

Appendix A: Case organisation

The data of the CFD study were located mainly in the local folder */scratch/mb483*.

Different folders are within it: *Bis3*, *Bis4*, *mypimplefoam2*, *pressuretests*, *Wind* folders, *Windtests* folders and *Windstored tests*.

Each of *Bis3* and *Bis4* contain five folders corresponding to the cases of the different patients named: *P1*, *P2*, *P3*, *H1* and *H2*.

The letter P is for unhealthy patients and the letter H is for healthy patients.

mypimplefoam2 contains the modified version of *pimplefoam* that was used when implementing the hard-coded windkessel boundary condition.

pressuretests is constituted of the different test-cases that were used to locate a mistake (which has been corrected later) that gave unreliable results.

The Wind folders contain the hard-coded version of the windkessel boundary condition. The number next to the name of the folder indicates the version of the code. Each version of the test was associated with different tests that are within the *Windtests* folders numbered accordingly.

Windstored tests was the folder in which the test-cases with the groovyBC windkessel boundary condition was stored.

The ScanIP files were also stored in the following directory: *OpenFOAM/mb483-2.1.0/run/ScanIP*

Appendix B: Healthy patients' circle of Willis

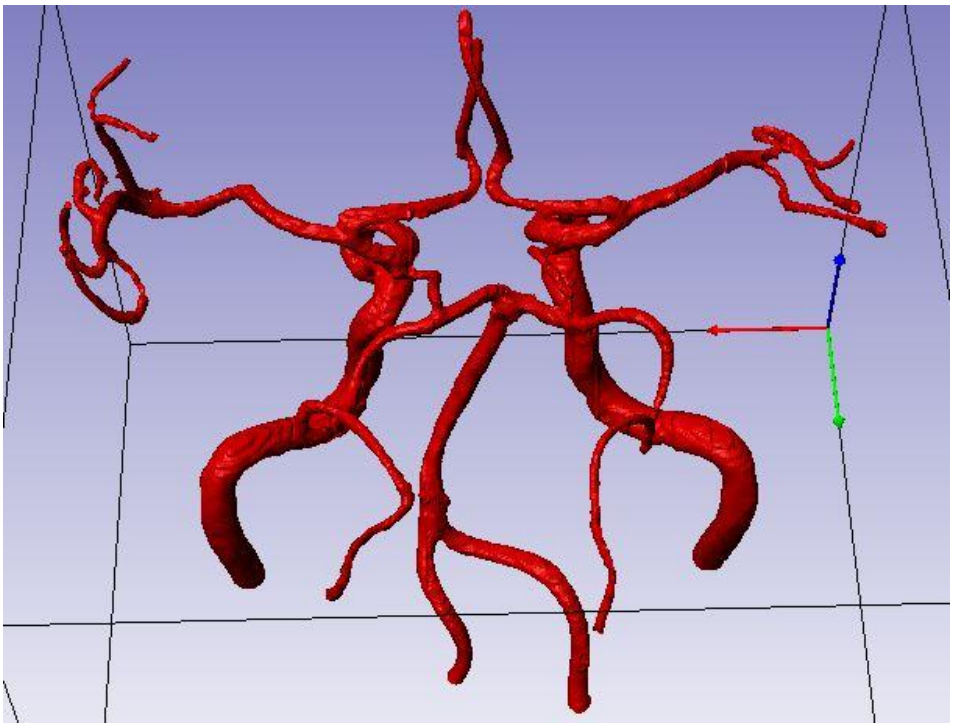

Figure 43 Circle of Willis of the first Healthy patient

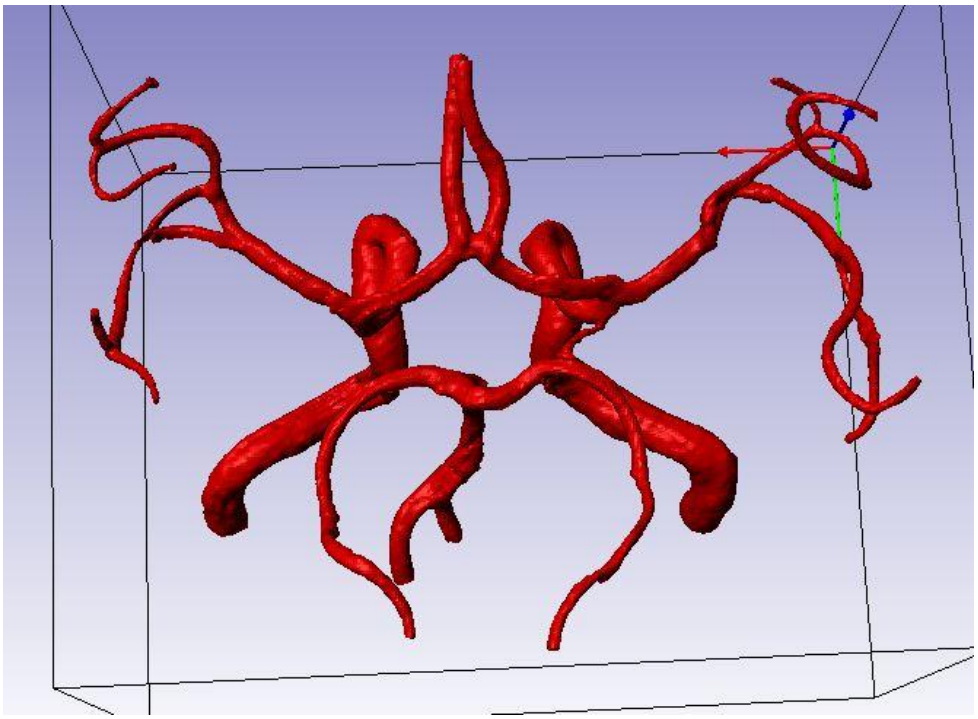

Figure 44 Circle of Willis of the second healthy patient

Appendix C: list of figures

Appendix D: list of tables

Appendix E: Windkessel BC with stored values

```
    outlet
    {
      type            groovyBC;
      storedVariables      (
      {
           name    pstored;
           initialValue        "0";
           }

      {
           name    phistored;
           initialValue        "0";
           }
           );

      variables
"R1=1;R2=1;C=1;pv=0;a1=R2*C/(R2*C+deltaT());a2=deltaT()/R2*C;pstored=a1*(pstored+a2*(pv+(R1
+R2)*sum(phi)+R1*R2*C*(sum(phi)-phistored)/deltaT()));phistored=sum(phi);";
      valueExpression "pstored";
      value           $internalField;
    }
```

Appendix F: Hard-coded windkessel BC

a. Header

```
#ifndef Windkessel7FvPatchScalarField_H
#define Windkessel7FvPatchScalarField_H

#include "mixedFvPatchFields.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

/*---------------------------------------------------------------------------*\
                Class Windkessel7FvPatchScalarField Declaration
\*---------------------------------------------------------------------------*/

class Windkessel7FvPatchScalarField
:
    public mixedFvPatchField<scalar>
{
protected:

    // Private data

        //- Name of the flux transporting the field
        word phiName_;

        //- Name of the flux transporting the field
        word dphiName_;
        //- Name of the flux transporting the field
        word rhoName_;
        //- Name of the density field used to normalise the mass flux
        //  if neccessary
        scalar rhovalue_;

        //- Resistance R1
        scalar R1_;

        //- Resistance R2
        scalar R2_;

        //- Compliance C
        scalar C_;

        //- Pressure at the entrance of the venous system
        scalar pv_;


public:

    //- Runtime type information
    TypeName("Windkessel7");


    // Constructors
```

```
//- Construct from patch and internal field
Windkessel7FvPatchScalarField
(
    const fvPatch&,
    const DimensionedField<scalar, volMesh>&
);

//- Construct from patch, internal field and dictionary
Windkessel7FvPatchScalarField
(
    const fvPatch&,
    const DimensionedField<scalar, volMesh>&,
    const dictionary&
);

//- Construct by mapping given Windkessel7FvPatchScalarField
//  onto a new patch
Windkessel7FvPatchScalarField
(
    const Windkessel7FvPatchScalarField&,
    const fvPatch&,
    const DimensionedField<scalar, volMesh>&,
    const fvPatchFieldMapper&
);

//- Construct as copy
Windkessel7FvPatchScalarField
(
    const Windkessel7FvPatchScalarField&
);

//- Construct and return a clone
virtual tmp<fvPatchScalarField> clone() const
{
    return tmp<fvPatchScalarField>
    (
        new Windkessel7FvPatchScalarField(*this)
    );
}

//- Construct as copy setting internal field reference
Windkessel7FvPatchScalarField
(
    const Windkessel7FvPatchScalarField&,
    const DimensionedField<scalar, volMesh>&
);

//- Construct and return a clone setting internal field reference
virtual tmp<fvPatchScalarField> clone
(
    const DimensionedField<scalar, volMesh>& iF
) const
{
    return tmp<fvPatchScalarField >
    (
        new Windkessel7FvPatchScalarField(*this, iF)
    );
}
```

```
// Member functions

    // Access

        //- Return rho
        const scalar& rhovalue() const
        {
            return rhovalue_;
        }
        //- Return reference to rho to allow adjustment
        scalar& rhovalue()
        {
            return rhovalue_;
        }
        //- Return reference to R1 to allow adjustment
        scalar& R1()
        {
            return R1_;
        }
        //- Return R1
        const scalar& R1() const
        {
            return R1_;
        }

        //- Return R2
        scalar R2() const
        {
            return R2_;
        }

        //- Return reference to R2
        //  to allow adjustment
        scalar& R2()
        {
            return R2_;
        }

        //- Return C
        const scalar& C() const
        {
            return C_;
        }

        //- Return reference to C to allow adjustment
        scalar& C()
        {
            return C_;
        }
        //- Return pv
        const scalar& pv() const
        {
            return pv_;
        }

        //- Return reference to pv to allow adjustment
        scalar& pv()
        {
            return pv_;
        }
```

```
        // Evaluation functions

/*          //- Calculate and return the advection speed at the boundary
          virtual tmp<scalarField> advectionSpeed() const;*/

          //- Update the coefficients associated with the patch field
          virtual void updateCoeffs();


        //- Write
        virtual void write(Ostream&) const;
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

/*#ifdef NoRepository
#   include "Windkessel7FvPatchScalarField.C"
#endif*/

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

b. C file

```
/*---------------------------------------------------------------------------*\
 =========                 |
 \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
  \\    /   O peration     |
   \\  /    A nd           | Copyright (C) 2011 OpenFOAM Foundation
    \\/     M anipulation  |
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/
```

```
#include "Windkessel7FvPatchScalarField.H"
#include "addToRunTimeSelectionTable.H"
#include "fvPatchFieldMapper.H"
#include "volFields.H"
#include "EulerDdtScheme.H"
#include "CrankNicholsonDdtScheme.H"
#include "backwardDdtScheme.H"


// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //

Foam::Windkessel7FvPatchScalarField::Windkessel7FvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF
)
:
    mixedFvPatchField<scalar>(p, iF),
    phiName_("phi"),
    dphiName_("dphi"),
    rhoName_("rho"),
    rhovalue_(1000),
    R1_(0.0),
    R2_(0.0),
    C_(0.0),
    pv_(0.0)
{
    this->refValue() = pTraits<scalar>::zero;
    this->refGrad() = pTraits<scalar>::zero;
    this->valueFraction() = 0.0;
}


Foam::Windkessel7FvPatchScalarField::Windkessel7FvPatchScalarField
(
    const Windkessel7FvPatchScalarField& ptf,
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF,
    const fvPatchFieldMapper& mapper
)
:
    mixedFvPatchField<scalar>(ptf, p, iF, mapper),
    phiName_(ptf.phiName_),
    dphiName_(ptf.dphiName_),
    rhoName_(ptf.rhoName_),
    rhovalue_(ptf.rhovalue_),
    R1_(ptf.R1_),
    R2_(ptf.R2_),
    C_(ptf.C_),
    pv_(ptf.pv_)
{}


Foam::Windkessel7FvPatchScalarField::Windkessel7FvPatchScalarField
(
    const fvPatch& p,
    const DimensionedField<scalar, volMesh>& iF,
    const dictionary& dict
)
:
```

```
    mixedFvPatchField<scalar>(p, iF),
    phiName_(dict.lookupOrDefault<word>("phi", "phi")),
    dphiName_(dict.lookupOrDefault<word>("dphi", "dphi")),
    rhoName_(dict.lookupOrDefault<word>("rho", "rho")),
    rhovalue_((dict.lookupOrDefault<scalar>("rhovalue", pTraits<scalar>::zero))),
    R1_((dict.lookupOrDefault<scalar>("R1", pTraits<scalar>::zero))),
    R2_((dict.lookupOrDefault<scalar>("R2", pTraits<scalar>::zero))),
    C_((dict.lookupOrDefault<scalar>("C", pTraits<scalar>::zero))),
    pv_((dict.lookupOrDefault<scalar>("pv", pTraits<scalar>::zero)))
{
    if (dict.found("value"))
    {
        fvPatchField<scalar>::operator=
        (
            Field<scalar>("value", dict, p.size())
        );
    }
    else
    {
        fvPatchScalarField::operator=(this->patchInternalField());
    }

    this->refValue() = *this;
    this->refGrad() = pTraits<scalar>::zero;
    this->valueFraction() = 0.0;

}


Foam::Windkessel7FvPatchScalarField::Windkessel7FvPatchScalarField
(
    const Windkessel7FvPatchScalarField& ptpsf
)
:
    mixedFvPatchField<scalar>(ptpsf),
    phiName_(ptpsf.phiName_),
    dphiName_(ptpsf.dphiName_),
    rhoName_(ptpsf.rhoName_),
    rhovalue_(ptpsf.rhovalue_),
    R1_(ptpsf.R1_),
    R2_(ptpsf.R2_),
    C_(ptpsf.C_),
    pv_(ptpsf.pv_)
{}


Foam::Windkessel7FvPatchScalarField::Windkessel7FvPatchScalarField
(
    const Windkessel7FvPatchScalarField& ptpsf,
    const DimensionedField<scalar, volMesh>& iF
)
:
    mixedFvPatchField<scalar>(ptpsf, iF),
    phiName_(ptpsf.phiName_),
    dphiName_(ptpsf.dphiName_),
    rhoName_(ptpsf.rhoName_),
    rhovalue_(ptpsf.rhovalue_),
    R1_(ptpsf.R1_),
    R2_(ptpsf.R2_),
    C_(ptpsf.C_),
```

```
    pv_(ptpsf.pv_)
{}


// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

void Foam::Windkessel7FvPatchScalarField::updateCoeffs()
{
    if (this->updated())
    {
        return;
    }

    word ddtScheme
    (
        this->dimensionedInternalField().mesh()
       .ddtScheme(this->dimensionedInternalField().name())
    );
    scalar deltaT = this->db().time().deltaTValue();

    const GeometricField<scalar, fvPatchField, volMesh>& field =
        this->db().objectRegistry::lookupObject<GeometricField<scalar, fvPatchField, volMesh> >
        (
            this->dimensionedInternalField().name()
        );

    const scalar w=R2_*C_/(R2_*C_+deltaT);


    const scalar alpha=deltaT/(R2_*C_*rhovalue_);

    label patchi = this->patch().index();

    fvsPatchField<scalar> phip =
        this->patch().lookupPatchField<surfaceScalarField, scalar>
        (
            phiName_
        );

    fvsPatchField<scalar> dphip =
        this->patch().lookupPatchField<surfaceScalarField, scalar>
        (
            dphiName_
        );

        if
        (
            ddtScheme == fv::EulerDdtScheme<scalar>::typeName
         || ddtScheme == fv::CrankNicholsonDdtScheme<scalar>::typeName
        )
        {

            this->refValue() =
            (

w*(field.oldTime().boundaryField()[patchi]+alpha*(pv_+(R1_+R2_)*gSum(phip)+R1_*R2_*C_*gSum(d
phip)))
            );

            this->valueFraction() = 1;
```

```
        }
        else if (ddtScheme == fv::backwardDdtScheme<scalar>::typeName)
        {

            this->refValue() =
            (

w*(field.oldTime().boundaryField()[patchi]+alpha*(pv_+(R1_+R2_)*gSum(phip)+R1_*R2_*C_*gSum(d
phip)))
            );

            this->valueFraction() = 1;
        }
        else
        {
            FatalErrorIn
            (
                "Windkessel7FvPatchScalarField::updateCoeffs()"
            )   << "    Unsupported temporal differencing scheme : "
                << ddtScheme
                << "\n    on patch " << this->patch().name()
                << " of field " << this->dimensionedInternalField().name()
                << " in file " << this->dimensionedInternalField().objectPath()
                << exit(FatalError);
        }

    mixedFvPatchField<scalar>::updateCoeffs();
}


void Foam::Windkessel7FvPatchScalarField::write(Ostream& os) const
{
    fvPatchField<scalar>::write(os);

    if (phiName_ != "phi")
    {
        os.writeKeyword("phi") << phiName_ << token::END_STATEMENT << nl;
    }
    if (phiName_ != "dphi")
    {
        os.writeKeyword("dphi") << phiName_ << token::END_STATEMENT << nl;
    }
    if (rhoName_ != "rho")
    {
        os.writeKeyword("rho") << phiName_ << token::END_STATEMENT << nl;
    }


    this->writeEntry("value", os);
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
    makePatchTypeField
    (
        fvPatchScalarField,
        Windkessel7FvPatchScalarField
    );
```

```
}

// ********************************************************************** //
```

References

[1] TJ Pedley, *The fluid mechanics of large blood vessels*, Cambridge University press, 1980

[2] Caro, Pedley, Schroter, Seed, *The mechanics of circulation*, Oxford University press, 1978

[3] E.O. Attinger, *Pulsatile blood flow*, McGraw Hill book company, 1964

[4] Wen-Jei Yang, *Biothermal fluid-sciences*, Hemisphere publishing corporation, 1989

[5] J. Alastruey, K.H. Parker, J. Peiró, S.M. Byrd, S.J. Sherwin, *Modelling the circle of Willis to assess the effects of anatomical variations and occlusions on cerebral flows*, Journal of Biomechanics, Volume 40, Issue 8, 2007, Pages 1794–1805.

[6] Marie Willemet, Valérie Lacroix, Emilie Marchandise, *Inlet boundary conditions for blood flow simulations in truncated arterial networks*, Journal of Biomechanics, Volume 44, Issue 5, 15 March 2011, Pages 897-903, ISSN 0021-9290, 10.1016/j.jbiomech.2010.11.036.

[7] João Janela, Alexandra Moura, Adélia Sequeira, *Absorbing boundary conditions for a 3D non-Newtonian fluid–structure interaction model for blood flow in arteries*, International Journal of Engineering Science, Volume 48, Issue 11, November 2010, Pages 1332-1349, ISSN 0020-7225, 10.1016/j.ijengsci.2010.08.004.

[8] Irene E. Vignon-Clementel, C. Alberto Figueroa, Kenneth E. Jansen, Charles A. Taylor, *Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries*, Computer Methods in Applied Mechanics and Engineering, Volume 195, Issues 29–32, 1 June 2006, Pages 3776-3796, ISSN 0045-7825, 10.1016/j.cma.2005.04.014.

[9] Julia Mikhal, Bernard J. Geurts, *Pulsatile flow in model cerebral aneurysms*, Procedia Computer Science, Volume 4, 2011, Pages 811-820, ISSN 1877-0509, 10.1016/j.procs.2011.04.086.

[10] W. Cousins, P.A. Gremaud, *Boundary conditions for hemodynamics: The structured tree revisited*, Journal of Computational Physics, Volume 231, Issue 18, 15 July 2012, Pages 6086-6096, ISSN 0021-9991, 10.1016/j.jcp.2012.04.038.

[11] Shigekazu Takeuchi, Takeshi Karino, *Flow patterns and distributions of fluid velocity and wall shear stress in the human internal carotid and middle cerebral arteries*, World Neurosurgery, Volume 73, Issue 3, March 2010, Pages 174-185, ISSN 1878-8750, 10.1016/j.surneu.2009.03.030.

[12] Alvaro Valencia, Hernan Morales, Rodrigo Rivera, Eduardo Bravo, Marcelo Galvez, *Blood flow dynamics in patient-specific cerebral aneurysm models: The relationship between wall shear stress and aneurysm area index*, Medical Engineering & Physics, Volume 30, Issue 3, April 2008, Pages 329-340, ISSN 1350-4533, 10.1016/j.medengphy.2007.04.011.

[13] Martin Sandve Alnæs, Jørgen Isaksen, Kent-André Mardal, Bertil Romner, Michael K. Morgan, Tor Ingebrigtsen, *Computation of Hemodynamics in the Circle of Willis*, *Stroke. 2007;38:2500-2505, published online before print August 2 2007, doi:10.1161/STROKEAHA.107.482471*

[14] N. Westerhof, J-W. Lankhaar, B. Westerhof, *The arterial Windkessel*, Medical and Biological Engineering and Computing, 2009-02-01, Springer Berlin / Heidelberg, Issn: 0140-0118, Page: 131-141, Volume 47, Issue 2, doi: 10.1007/s11517-008-0359-2

[15] Daniel R. Kerner, Solving Windkessel Models with MLAB, http://www.civilized.com/mlabexamples/windkesmodel.htmld/

[16] How to add a new boundary condition, http://openfoamwiki.net/index.php/HowTo_Adding_a_new_boundary_condition

[17] CFD-online, http://cfd-online.com

[18] H. Harvey, *Modelling Blood Flow Through the Circle of Willis: A Study Combining Image-Based Meshing of MRI Data with CFD*, (unpublished paper), University of Exeter , 2010

[19] S. Daniels, *The Computational Investigation of the Blood Flow in the Circle of Willis*, (unpublished paper), University of Exeter,2011

[20] Moore. S.M, David T, Chase JG, Arnold J, Fink J. (2006) *3D models of blood flow in the cerebral vasculature*. Journal of Biomechanics. 39:1454–1463

[21] R. Himeno, *Blood Flow Simulation toward Actual Application at Hospital*, The 5th Asian Computational Fluid Dynamics, Busan, Korea, October 27th - October 30th , 2003