Politecnico di Milano

IV Facoltà di Ingegneria Industriale

Corso di Laurea Specialistica in Ingegneria Aeronautica

Dipartimento di Ingegneria Aerospaziale (DIA)
Dipartimento di Matematica "F. Brioschi"
MOX - Modeling & Scientific Computing

"Nonlinear Fluid-Structure interaction with applications in computational haemodynamics"

Relatore: Prof. Luca FORMAGGIA
Co-relatore: Dr. Fabio NOBILE

Autore:
Gianmarco MENGALDO 740313

Anno accademico 2010/2011

A papà,
tra ponti, calli
e campielli

Ringraziamenti

Desidero ringraziare il prof. Luca Formaggia che mi ha fatto scoprire l'affascinante mondo dell'analisi numerica fin dai primi anni di università con il corso di "Calcolo Numerico" e con "Modellistica numerica per problemi differenziali" poi. Durante questa esperienza di tesi, ha saputo darmi le giuste indicazioni dal punto di vista professionale e un continuo sostegno dal punto di vista umano di cui non dubitavo. Ringrazio il dr. Fabio Nobile per le lunghe ed esaustive "chiacchierate" sugli aspetti implementativi e modellistici del lavoro. Ringrazio inoltre Mariarita de Luca per i consigli riguardanti l'aspetto strutturale e per l'ospitalità presso la SISSA di Trieste, Paolo Tricerri per le discussioni e il confronto produttivo sulla parte di interazione fluido-struttura e Matteo Pozzoli per avermi introdotto a "LifeV" e agli aspetti tecnici del calcolo parallelo, nonchè per l'ospitalità sul cluster Lagrange del CILEA.

Un ringraziamento particolare va a mia madre che con i suoi sacrifici durante questi (quasi) 6 anni mi ha permesso di studiare a Milano.

Ringrazio inoltre tutti i compagni di viaggio conosciuti in questi anni universitari, Alex e Nicolò, con i quali ho passato un agosto a preparare l'Esame, Dani, Giulio, Attilio, Antonio e Gigi per le cene insieme, e l'amico Valerione. Non posso poi dimenticare gli amici del gruppo di Pavesi e Parigini, Fabiagiobio e Valeria e gli ex-coinquilini di viale Monza 44.

Infine ringrazio Vale e in modo particolare Abi per avermi sopportato e supportato in questi anni intensi e per avermi aiutato a superare i momenti difficili.

Abstract

The principal interest of this thesis is in computational haemodynamics and in particular in the interaction between non-linear isotropic arterial wall-mechanics and blood flow. The starting point of this work is therefore to correctly describe the mechanical behavior of the artery wall in regime of finite deformations with the nonlinear structural models commonly used to describe biological tissues, ie hyperelastic materials. Subsequently, the nonlinear structural models are coupled using a partitioned (segregated) strategies to the fluid equations.
All of the present work is developed with the Free Software object-oriented with license LGPL called *LifeV* that works on parallel architecture. In particular, the software is a finite element library that solves several physical problems, such as fluid dynamics, reaction-diffusion-transport and mechanical problems in a multiphysics contest. In particular the algorithms developed, simulate the three-dimensional, time-varying fluid-structure interaction (FSI) problems and the nonlinear three-dimensional, time-varying mechanical problems.

Keywords: Nonlinear fluid-structure interaction, hyperelasticity, haemodynamics, parallel computing, nonlinear structural models, large deformations.

Sommario

I principali interessi di questa tesi sono relativi all'emodinamica computazionale ed in particolare all'interazione tra la meccanica della parete arteriosa, modellata come materiale nonlineare isotropo, e il flusso sanguigno. Il punto di partenza di questo lavoro è stato quindi quello di descrivere correttamente il comportamento della parete di un'arteria in regime di deformazioni finite con modelli strutturali nonlineari tipicamente utilizzati per modellare tessuti biologici, quali sono i materiali iperelastici. Successivamente si è provveduto ad accoppiare tali modelli con un solutore fluido-struttura partizionato (segregato).

Il presente lavoro è stato interamente sviluppato con un Software orientato a oggetti, libero con licenza LGPL chiamato *LifeV* in grado di lavorare in parallelo. Il Software è una libreria ad elementi finiti scritta in linguaggio C++ che risolve diversi problemi fisici quali ad esempio, problemi in ambito fluidodinamico, meccanico, di diffusione-trasporto-reazione in un contesto multiscala. In particolare, gli algoritmi sviluppati modellano problemi di interazione fluido-struttura e di meccanica nonlineare, tridimensionali e tempo-varianti.

Parole chiave: Interazione fluido-struttura nonlineare, iperelasticità, emodinamica, calcolo parallelo, modelli strutturali nonlineari, grandi deformazioni.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and motivations

The chapter is structured in four parts. The first one presents the motivations and targets of this work with a general overview of the more common cardiovascular diseases and how they may relate to haemodynamic factors. The second part considers the state-of-the-art of the mechanical description of an artery and the state-of-the-art of fluid-structure interaction (FSI) in haemodynamics. The third part presents the contribution of this work in the mechanical and haemodynamic fields. The last section presents the outline of this thesis.

1.1 Motivations and goals

The development of increasingly accurate simulation tools in hemodynamics, allows to better understand the phenomena at the basis of development of cardiovascular diseases (CVD) such as atherosclerotic plaques, blood clots and aneurysms. In recent years substantial progress has been made in this field of research, which, in the long terms should be able to better describe the patologies mentioned, but at the same time provide tools for accurate diagnosis and prevention. In perspective, it may be able to provide guidance to medical intervention, such as the exclusion of an aneurysm or the application of a stent.

The growing interest in the simulations of the cardiovascular system is mainly linked to the considerable resources the cure of the cardiovascular diseases require from the health system. There is certainly a need to a change of lifestyle to prevent diseases induced, for example, by smoking or an incorrect diet, such as heart attack or atherosclerosis. At the same time, however, it is necessary to provide effective investigative tools to treat existing diseases and better understand the phenomena behind this type of problems and provide a more effective prevention. To understand the size of the problem, it is enough to recall that CVD is the leading cause of death in the industrialised world. About 30% of deaths in fact, happens for such diseases. In 2010, 18.1 million people died of CVD. In particular, stroke is the second leading cause of death globally, and the leading cause of acquired disability, killing 5.7 million people every year, of whom 85% live in the developing world. Today, 1.3 billion people smoke worldwide, 600 million have hypertension, and 220 million live with diabetes which puts more than 2 billion individuals at risk of heart disease, stroke, or a related health

problem [2]. It has been estimated that only in USA, the medical costs of CVD in 2010 amounted to 171 billion dollars. By 2030 it is estimated a cost increase of 61% to reach the level of 275.8 billion dollars [3].

Also in European Union (EU), the CVD medical cost are very relevant. The financial charge on EU health systems was estimated at just under 110 billion € in 2006. This equates to a cost per person of 223 € per year, approximately 10% of total healthcare expenditure in the EU [4].

The interest of this thesis, is aimed to the correct description of the behavior of an artery. Specifically, we want to describe more accurately the mechanical behavior of the arterial wall, using nonlinear structural models, and their mutual influence with the fluid-dynamic field of the blood.

In particular, we have considered three models: St.Venant-Kirchhoff, Neo-Hookean and Exponential. The reason for the choice is due to the fact that these models have some properties similar to biological tissues as well known in literature [5], [6], [7] and [8]. Moreover, the last two models mentioned above could be combined with a multi-mechanism approach (briefly described in the next section) to grasp more accurately the structural mechanics of an artery and the first stages of formation of a aneurysm. Finally, the description of the FSI problem with nonlinear structural models allows to have haemodynamic indices more precise with respect to the use of linear model.

Summarizing, the principal motivation and perspectives of this work are the following:

1. Better understanding of the structural dynamics behaviour of arteries.
2. Development of fexible and efficient algorithms that can be reused in other contest different than haemodynamics.
3. Development of the efficient tools be able to describe FSI problem with large added-mass effect using nonlinear structural models.
4. Development of a prototype code that constitutes a first step toward a complete tool for diagnosis and prevention in cardiovascular field.

The work carried out in this thesis is a first step in this direction and concentrates most on issues 1 and 2.

1.2 State of the art

Artery wall mechanics

Arteries are subdivided into two types, *elastic* and *muscular*. The first one presents an elastic behaviour while the second type is characterized by an important viscoelastic effect [7]. Another important characteristics of the passive mechanical behaviour of an artery is that the stress-strain response during loading and unloading stages is highly non-linear. Infact

at low strain levels the structural response is mainly governed by *elastina* (soft tissue) while at high levels of strain the structural response is governed by fibers of collagen that stiffen the behaviour of the wall's artery. Moreover, in large strain, the mechanical beahaviour is anisotropic because of the collagen fibers structure [5].

The arterial wall is divided into three different layers: *tunica intima*, *tunica media* and *tunica externa* as shown in figure 1.1. The first layer is the innermost, and does not contribute to the passive wall-mechanical response in the young arteries [6], although, in the pathological case (i.e. atherosclerosis), its contribution is very important. The second layer is the most important for arterial structural behaviour because in this part there are the *elastin* and the *collagen* that are the key in the description of artery wall-mechanics. Finally the third arterial layer is the outermost and has a small contribution to structural response [9].

Actually, some research groups are oriented towards a description of the arterial wall that

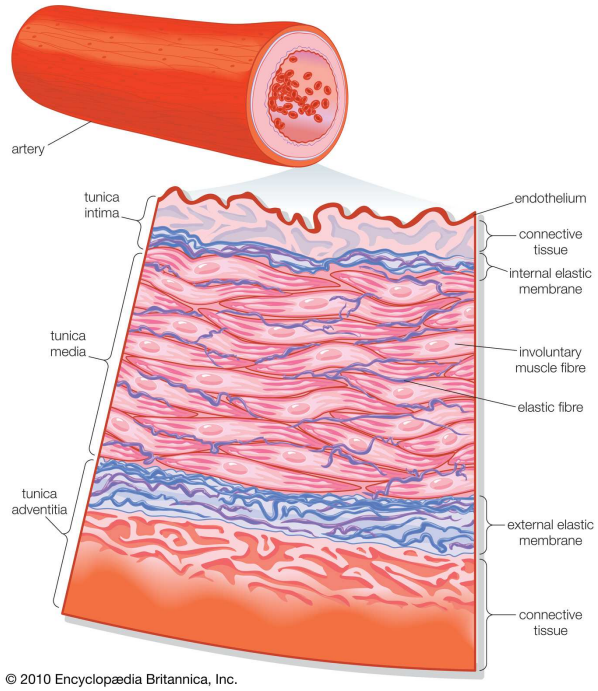


Figure 1.1: Schematic representation of the artery wall [1]

takes into account a multi-mechanism, that of the elastin and collagen [10], [11],[12]. The multi-mechanism models are based on the physiological assumption that the elastin works at low strain levels, then enters into a region of deformation in which collagen and elastin work together, and finally enters into a region of deformation in which only collagen works.

As anticipated, elastin can be considered an isotropic material with Young's modulus of about 1.1 MPa, while collagen is composed of fibers that make it anisotropic and has a stiffness greater than elastin with a Young's modulus of about 1.1GPa. From a computational point of view, this behavior can be effectively described using two different nonlinear constitutive laws, the first working under a certain threshold of strain and the second working above it.

Furthermore, with regard to collagen, it is possible to build a model that includes anisotropic behaviour.

The first interest of this thesis is to implement the typical material models that describe the biological tissues like arterial wall in regime of finite deformations and, at the same time, to couple these models to the fluid equations. In particular, the hyperelastic material models are tipically used to describe soft-biological tissues. The peculiarity of these types of models is the nonlinear relation between stress and strain. Moreover, the request of finite deformations is a good starting-point to describe one of the most important pathological cases that affectes the arteries: aneurysm.

This work, is thus the basis for more complex models to describe the real behavior of the wall of an artery subjected to large deformations and is able itself to provide more precise information on the status of stress and deformation in the wall than the model of linear elasticity.

FSI in haemodynamic problems

Fluid-structure interaction problems (FSI) is very important in many engineering situations like vibrations of aeronautics structures and suspended bridge, flow into pipelines, oscillations of long electrical cables' spans and many others. In the last years FSI has developed a relevant attention even in the life science context.

FSI problems can be modeled using two different approaches: monolithic or partitioned (segregated) procedures [13]-[14]. In the monolithic approach the fluid and structural equations are solved simultaneously, while in the partitioned procedures the fluid and structural equations are solved separately. Monolithic approach is a stable method while the partitioned approach may be unstable but it is more flexible because it allows to reuse possible existing algorithms and codes for the structural and fluid problems.

The numerical solution of FSI problems is very complex due to the nonlinearity present in the constitutive equations and induced by the coupling. In addition to these, within hemodynamics, it is necessary to pay close attention to the added mass effect that occurs when the density of the fluid and solid are similar [15]. In fact, in this case it is possible to notice a deterioration in the convergence properties of the numerical problem. Some techniques are used to improve the convergence properties of the algorithms and to reduce the computational costs to solve an FSI problem. In particular, for partitioned approaches, Robin-Neumann and Robin-Robin transmission conditions allow to reduce the number of iterations per time-step to reach convergence [16],[17], [18].

In many works, the structure problem is described by a linear model which is not true in this work. In fact, the nonlinearities of the FSI problem are added to the nonlinearity of the structural part. This implies an increase of computational costs and a possible deterioration of the convergence of the overall problem. It is clear therefore, the need of using algorithms that are efficient enough and reduce the deterioration of the convergence properties caused by the added mass. To solve the FSI problem a partitioned interface Newton Krylov method [19], [20], [21], [22] has been used in this work, which has a good computational efficiency compared to other approaches such as fixed-point methods with relaxation [23]. In addition,

the solution of FSI problems with the use of partitioned algorithms, allows easier implementation of individual sub-problems, enabling to implement state of the art algorithms for the structural and fluid dynamics problems. This last point is very important because the nonlinear structural solver should be further manipulated to introduce more complex models. It is therefore necessary to have an easy way to insert it into an FSI solver.

1.3 Contribution of the thesis

In the first part of this thesis, we have implemented a *parallel*[1] structural (mechanical) solver into the open-source finite-element library *LifeV*, starting from a pre-existent algorithm. In particular, we have added to existing material, St. Venant-Kirchhoff, two nonlinear materials typically used for biological tissues: Neo-Hookean model and Exponential model. St. Venant-Kirchhoff material is a hyperelastic compressible material, while the other two materials, Neo-Hookean and Exponential, are nearly-incompressible hyperelastic materials [24]-[25]-[26]-[27]. Therefore, the mechanical solver that we have implemented, is able to solve mechanical problems in regime of finite deformations using three different nonlinear constitutive laws:

- St. Venant-Kirchhoff;
- Neo-Hookean;
- Exponential.

We have then validated the solver from a quantitative point of view, using a simple academic test case of uniaxial traction on a cube. Moreover, we have done some test cases of inflation of an hollow cylinder. In this case, a closed form of the exact solution is not available for the three nonlinear structural models implemented. However, we can compare, from a qualitative point of view, the solution obtained using nonlinear structural models with the solution obtained using linear material for which an analytical solution exists. In this way, we can see if the results are reasonable in modulus and shape. Moreover, we have seen if the solution is symmetric as expected, evaluating the radial displacement for each node around the circumference of a certain section of the hollow cylinder. Finally, we have done a mesh convergence test for all materials.

In the second part of this thesis, we have coupled the mechanical solver with a fluid solver. In particular, we have integrated the structural solver in a *parallel* partitionated FSI solver. We have done some simulations, using a simple geometry (straight cylinder) and the different structural models (including linear structural model) to evaluate the correctness of the results from a qualitative point of view, as again the exact solution is not known.

Completed preliminary testing, we have evaluated the performance of mechanical and FSI codes. In particular we have evaluated the residual vs. Newton iterations, the number of Newton iterations per time-step, the dimensionless CPU-time to perform the main subroutines and to complete a standard simulation using the different structural models.

[1] with the term *parallel* we mean that the algorithm works in a parallel architecture

Finally, we have performed a simulation on a geometry of the carotid artery. In this case, we have evaluated the fluid velocity field, the fluid pressure and the structure displacement at certain sections of the carotid. Moreover, we have evaluated the wall-shear stress, a parameter tipically used in haemodynamics, and we have compared the results obtained using the different materials.

Summarizing, the present work has produced the following results:

- Development of a parallel nonlinear mechanical solver.
- Integration of the nonlinear mechanical solver in a parallel FSI solver.
- Creation of some academic test cases (with related documentation) to evaluate the correctness of the results of the mechanical and FSI solvers.
- Haemodynamics results on the carotid arteries.

1.4 Outline of the thesis

In Chapter 2, we briefly introduce three-dimensional finite elasticity and the structural models we have implemented. Moreover, the finite element formulation, time discretization and design of the structural solver are outlined and we present the numerical validation of the structural solver on simple test cases.

In Chapter 3, we briefly introduce the FSI problem and present the algorithm adopted and implemented in this work. To show the effectiveness of the algorithm we present an ideal test case on a straight cylinder and compare qualitatively the solution obtained with linear elasticity and nonlinear structural models.

In Chapter 4, we provide some implementation details and discuss the performances of the structural solver and FSI solver. In particular we show the correct parallelization of the structural solver with a test of scalability on the Lagrange cluster of CILEA.

In Chapter 5, we consider an haemodynamic problem on real carotid geometry. In particular we emphasize the differences between linear and nonlinear structural models in haemodynamic indices and in the fluid velocity, fluid pressure and structural displacement at certain sections of the carotids.

In Chapter 6, we comment critically the results obtained in this work and we suggest possible future developments.

Chapter 2

Nonlinear structural model

One of the purposes of this work is to implement nonlinear models of isotropic materials in the FE (finite-element) library LifeV. In particular we have created a class to implement three constitutive laws: St. Venant-Kirchhoff model, Neo-Hookean model and Exponential model. In the present chapter, three-dimensional finite elasticity is introduced and we describe the implementation of the three constitutive laws in LifeV. Moreover we provide a validation of the code on simple test cases. We remark that the code works on parallel architecture, that why we sometimes indicate it by *LifeV-parallel* to emphazise this peculiarity.

2.1 Three-dimensional finite elasticity

2.1.1 Kinematics

Three-dimensional finite elasticity describes the behaviour of a continuous body β in terms of kinematics, stress and deformation states. The body is assumed to occupy a compact domain in the three-dimensional Euclidean space, denoted by $\mathbb{E}$. β is made of material points whose position relative to a generic observer O at a time t defines the configuration of the body itself.

A configuration is a smooth-mapping of β onto a region of $\mathbb{E}$. It is possible to define one configuration that is constant in time, namely the *reference configuration* which describes the position of each material point with respect to its position in the reference configuration. The reference configuration is also called undeformed configuration, while the current configuration is also called deformed configuration. Furthermore, we assume that the reference configuration is at its natural state, that is the Cauchy stresses are everywhere zero. A transformation from undeformed configuration denoted by $\widehat{\Omega}$ to current configuration, denoted by Ω, is defined by a one-to-one, orientation-preserving vector field $\mathcal{L}$:

$$\begin{aligned} \mathbf{x} &= \mathcal{L}(\widehat{\mathbf{x}}, t)\,, \\ \widehat{\mathbf{x}} &= \mathcal{L}^{-1}(\mathbf{x}, t)\,. \end{aligned} \tag{2.1}$$

where $\mathbf{x} \in \Omega$ and $\widehat{\mathbf{x}} \in \widehat{\Omega}$.

The position of a material point in the reference configuration is denoted by $\widehat{\cdot}$ (i.e. $\widehat{\mathbf{x}}$). Furthermore, the differential operators and the element of area or volume are indicated by $\widehat{\cdot}$ (i.e. $\widehat{\nabla}$, $\widehat{\partial\Omega}$, $\widehat{\Omega}$) when they are referred to the reference configuration.

Remark 1. When it is clear from the context that a quantity belong to the reference configuration we omit $\widehat{\cdot}$.

Using the transformation from reference configuration to the current configuration, (2.1), the displacement field $\boldsymbol{\eta}$ in the reference configuration is defined as[1]:

$$\boldsymbol{\eta}(\widehat{\mathbf{x}}) = \mathcal{L}(\widehat{\mathbf{x}}) - \widehat{\mathbf{x}}\,. \tag{2.2}$$

Moreover it is possible to define a primary measure of deformation $\mathbf{F}$ called deformation gradient, as:

$$\mathbf{F} = \widehat{\nabla}\,\mathcal{L} \;\text{ componentwise: }\; \mathrm{F}_{ij} = \frac{\partial \mathcal{L}_i}{\partial \widehat{x}_j}\,.$$

In addition it is possible to introduce a symmetric positive-define tensor called right Cauchy-Green tensor denoted by $\mathbf{C}$ which measures the length of a vector $\delta\mathbf{x}$ after a generic deformation. In fact, a vector defined in the reference configuration $\delta\widehat{\mathbf{x}}$ that follows the material points is transformed within the first order into the vector $\delta\mathbf{x} = \mathbf{F}\delta\widehat{\mathbf{x}}$ and the length of $\delta\mathbf{x}$ is given by: $|\delta\mathbf{x}|^2 = \delta\widehat{\mathbf{x}}^{\mathrm{T}}(\mathbf{F}^{\mathrm{T}}\mathbf{F})\delta\widehat{\mathbf{x}}$. Hence it is convenient to introduce the right Cauchy-Green tensor:

$$\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}\,. \tag{2.3}$$

It is important for the following analisys to define the principal invariants, $\mathrm{I}_1(\mathbf{C})$, $\mathrm{I}_2(\mathbf{C})$, $\mathrm{I}_3(\mathbf{C})$ of $\mathbf{C}$; they are necessary to analyze homogeneus pure strain and to describe the hyperelastic constitutive laws. The three principal invariants are:

$$\begin{aligned} \mathrm{I}_1(\mathbf{C}) &= \mathrm{tr}(\mathbf{C})\,, \\ \mathrm{I}_2(\mathbf{C}) &= \frac{1}{2}\big[(\mathrm{tr}\mathbf{C})^2 - \mathrm{tr}(\mathbf{C}^2)\big]\,, \\ \mathrm{I}_3(\mathbf{C}) &= \det(\mathbf{C})\,. \end{aligned} \tag{2.4}$$

We also introduce the Green-Lagrange tensor $\mathbf{E}$ which is another useful measure of deformation:

$$\mathbf{E} = \frac{1}{2}\Big(\mathbf{C} - \mathbf{I}\Big)\,. \tag{2.5}$$

The Cauchy stress tensor $\mathbf{T}_{\mathrm{s}}$ represents the state of stress of the elastic body β in the current configuration Ω. It is a symmetric second order tensor and it depends on the position $\mathbf{x}$ and time t:

$$\mathbf{T} = \mathbf{T}_{\mathrm{s}}(\mathbf{x}, \mathrm{t})\,.$$

[1]For the displacement field in the reference configuration, we omit the superscript $\widehat{\cdot}$

Through the Piola tranformation it is possible to push back the state of stress of the elastic body β into the reference configuration $\widehat{\Omega}$:

$$\mathbf{P} = \mathrm{J}\mathbf{T}_\mathrm{s}\mathbf{F}^{-\mathrm{T}}\,, \tag{2.6}$$

where $\mathbf{P}$ is called the first Piola-Kirchhoff tensor, which indeed describes the stress state in the reference configuration and $\mathrm{J} = \det(\mathbf{F})$.

2.1.2 The equations of motion

It is possible to describe the equations of motion through the second law of dynamics. In fact it is well known that the rate of change of the linear momentum equals the sum of surface and volume forces. The equations of motion applied to an arbitrary volume V(t), with boundary ∂V(t) become:

$$\frac{\mathrm{D}}{\mathrm{Dt}}\int_{\mathrm{V(t)}} \rho\dot{\boldsymbol{\eta}}\,\mathrm{dV} = \int_{\mathrm{V(t)}} \rho\mathbf{b}\,\mathrm{dV} + \int_{\partial\mathrm{V(t)}} \mathbf{t}\,\mathrm{dS}\,, \tag{2.7}$$

where ρ is the density associated to the elastic body β and $\dot{\boldsymbol{\eta}}$ is the first time-derivative of the displacement. Defining the continuity of the mass:

$$\frac{\mathrm{D}\rho}{\mathrm{Dt}} + \rho\boldsymbol{\nabla}\cdot\dot{\boldsymbol{\eta}} = 0\,, \tag{2.8}$$

where $\frac{\mathrm{D}\rho}{\mathrm{Dt}} = \frac{\partial\rho}{\partial\mathrm{t}} + \dot{\boldsymbol{\eta}}\cdot\boldsymbol{\nabla}\rho$, it is possible to rewrite the equations of motion using the transport theorem and (2.8):

$$\int_{\mathrm{V(t)}} \rho\ddot{\boldsymbol{\eta}}\,\mathrm{dV} = \int_{\mathrm{V(t)}} \rho\mathbf{b}\,\mathrm{dV} + \int_{\partial\mathrm{V(t)}} \mathbf{t}\,\mathrm{dS}\,, \tag{2.9}$$

where $\ddot{\boldsymbol{\eta}}$ is the second time-derivative of the displacement. The substitution of the stress vector with its representation by means of the Cauchy stress tensor $\mathbf{T}_\mathrm{s}$ and the use of the Gauss theorem allows to write:

$$\int_{\mathrm{V(t)}} \rho\ddot{\boldsymbol{\eta}}\,\mathrm{dV} = \int_{\mathrm{V(t)}} \rho\mathbf{b}\,\mathrm{dV} + \int_{\mathrm{V(t)}} \nabla\cdot\mathbf{T}_\mathrm{s}\,\mathrm{dV}\,. \tag{2.10}$$

This equation describes the motion by Eulerian variables in the current configuration.

It is more convenient to rewrite the equations of motion in terms of Lagrangian variables. From this point of view it is necessary to relate an infinitesimal volume and an infinitesimal oriented surface in Ω(t) to their counterpart in the reference configuration. Using the Jacobian J of the deformation gradient and the Piola transformation we have:

$$\begin{cases} \mathrm{d}\Omega = \mathrm{J}\,\mathrm{d}\widehat{\Omega} \\ \mathbf{n}\mathrm{dS} = \mathrm{J}\mathbf{F}^{-\mathrm{T}}\widehat{\mathbf{n}}\mathrm{d}\widehat{\mathrm{S}}\,. \end{cases}$$

Using these relations it is possible to rewrite (2.7) in the reference configuration:

$$\int_{\widehat{\mathrm{V}}} \mathrm{J}\rho\,\ddot{\boldsymbol{\eta}}\,\mathrm{d}\widehat{\mathrm{V}} = \int_{\widehat{\mathrm{V}}} \mathrm{J}\rho\,\mathbf{b}\,\mathrm{d}\widehat{\mathrm{V}} + \int_{\partial\widehat{\mathrm{V}}} \mathrm{J}\mathbf{T}_\mathrm{s}\mathbf{F}^{-\mathrm{T}}\widehat{\mathbf{n}}\,\mathrm{d}\widehat{\mathrm{S}}\,. \tag{2.11}$$

The quantity $\mathrm{J}\ \mathbf{T}_\mathrm{s}\mathbf{F}^{-T}$ is the first Piola-Kirchhoff tensor. Hence the last equation becomes:

$$\int_{\widehat{\mathrm{V}}} \widehat{\rho}\,\ddot{\boldsymbol{\eta}}\,\mathrm{d}\widehat{\mathrm{V}} = \int_{\widehat{\mathrm{V}}} \widehat{\rho}\,\mathbf{b}\,\mathrm{d}\widehat{\mathrm{V}} + \int_{\partial\widehat{\mathrm{V}}} \mathbf{P}\,\widehat{\mathbf{n}}\,\mathrm{d}\widehat{\mathrm{S}}\,. \tag{2.12}$$

Finally, through the Gauss theorem, it is possible to transform the surface integral into the volume integral. The final integral expression of the equation of motion is the following:

$$\int_{\widehat{\mathrm{V}}} \widehat{\rho}\,\ddot{\boldsymbol{\eta}}\,\mathrm{d}\widehat{\mathrm{V}} = \int_{\widehat{\mathrm{V}}} \widehat{\rho}\,\mathbf{b}\,\mathrm{d}\widehat{\mathrm{V}} + \int_{\widehat{\mathrm{V}}} \widehat{\nabla}\cdot\mathbf{P}\,\mathrm{d}\widehat{\mathrm{V}}\,. \tag{2.13}$$

Which is valid for any $\widehat{\mathrm{V}} \subset \widehat{\Omega}$, therefore we may infer the differential equations of the linear momentum in the reference configuration:

$$\widehat{\rho}\,\ddot{\boldsymbol{\eta}} = \widehat{\rho}\,\mathbf{b} + \widehat{\nabla}\cdot\mathbf{P}\,. \tag{2.14}$$

The differential problem needs to be finalized by setting the initial and boundary conditions. The initial conditions reads:

$$\begin{aligned} \boldsymbol{\eta}(\widehat{\mathbf{x}},0) &= \boldsymbol{\eta}_0(\widehat{\mathbf{x}})\,, \\ \dot{\boldsymbol{\eta}}(\widehat{\mathbf{x}},0) &= \dot{\boldsymbol{\eta}}_0(\widehat{\mathbf{x}})\,, \end{aligned} \tag{2.15}$$

while, the most common boundary conditions are the following:

- Dirichlet conditions:
$$\boldsymbol{\eta}(\widehat{\mathbf{x}},\mathrm{t}) = \boldsymbol{g}\,, \qquad \widehat{\mathbf{x}} \in \widehat{\Gamma}_D\,, \tag{2.16}$$
where $\widehat{\Gamma}_D \subset \widehat{\partial\Omega}$.

- Neumann conditions:
$$\mathbf{P}\widehat{\mathbf{n}}(\widehat{\mathbf{x}},\mathrm{t}) = \boldsymbol{h}\,,\ \widehat{\mathbf{x}} \in \widehat{\Gamma}_N\,, \tag{2.17}$$
where $\widehat{\Gamma}_N \subset \widehat{\partial\Omega}$.

Moreover it is necessary to define the relation between the stress tensor and the kinematics variables to characterize the mechanical properties of the continuum body β. In particular it is necessary to define the constitutive law that characterizes the material model adopted. In the next paragraph, the hyperelastic materials are introduced from a general point of view.

2.1.3 Constitutive laws: hyperelastic materials

The law that relates the stress tensor $\mathbf{T}_\mathrm{s}$ and the kinematics variables is called constitutive law. This relation characterizes the mechanical properties of the continuum body β. A material is defined *elastic* if the stress tensor $\mathbf{P}$ depends on the position of the material points[2] $\mathbf{x}$ and the deformation gradient $\mathbf{F}$:

$$\mathbf{P} = \widetilde{\mathbf{P}}(\mathbf{x},\mathbf{F})\,. \tag{2.18}$$

[2]If we have to implement an anisotropic constitutive law, the dependence on $\mathbf{x}$ is necessary

When the previous relation is indipedent of the position of the material points $\mathbf{x}$ the material is called *homogeneous*. Finally a material is *hyperelastic* when it does not dissipate energy during cyclic homogeneous deformations:

$$\mathrm{W}_{\mathrm{cycle}} = \int_0^{\mathrm{T}} \int_{\widehat{\Omega}} \mathbf{P} : \dot{\mathbf{F}} \, \mathrm{d}\widehat{\Omega} \mathrm{dt} = 0 \,, \tag{2.19}$$

along any deformation characterized by $\mathbf{x}(\mathrm{t} = \mathrm{T}) = \mathbf{x}(\mathrm{t} = 0)$ at any point of β. To define an hyperelastic material it is common to introduce the strain-energy function $\overline{\mathrm{W}}$ that represents the amount of elastic energy locally stored in the body β during the deformation $\mathcal{L}$. The form of the strain-energy function characterizes a material from another one. In addition, for hyperelastic materials it is common to distinguish between compressible, nearly incompressible and incompressible materials.

Hyperelastic compressible materials

An admissible homogeneous deformation for compressible materials reads:

$$\mathcal{L}(\widehat{\mathbf{x}}, \mathrm{t}) = \mathbf{F}(\mathrm{t})\widehat{\mathbf{x}} + \mathbf{c}(\mathrm{t}) \,, \tag{2.20}$$

with the constraint J >0. With the change of variable $\mathrm{t} \rightarrow \mathbf{F}(\mathrm{t})$ the equation (2.19) becomes:

$$\mathrm{W}_{cycle} = \mathrm{vol}(\widehat{\Omega}) \int_{\mathbf{F}(0)}^{\mathbf{F}(\tau)=\mathbf{F}(0)} \tilde{\mathbf{P}}(\mathbf{F}) : \mathrm{d}\mathbf{F} = 0 \,. \tag{2.21}$$

This relation implies that it must exist a scalar function whose gradient is equal to the first Piola-Kirchhoff stress tensor $\mathbf{P}$. In particular this scalar function is exactly the strain-energy function $\overline{\mathrm{W}}$ previously defined. Thus it is possible to write:

$$\mathbf{P}(\mathbf{F}) = \frac{\partial \overline{\mathrm{W}}(\mathbf{F})}{\partial \mathbf{F}} \,. \tag{2.22}$$

The strain-energy function $\overline{\mathrm{W}}$ has to respect the axiom of frame indifference. In particular $\overline{\mathrm{W}}$ must be independent of rigid motion, since it depends only on the deformation $\mathcal{L}$. We can explain this concept introducing a generic rotation tensor $\mathbf{R}$. The axiom of frame indifference, in this case, reads:

$$\overline{\mathrm{W}}(\mathbf{R}\,\mathbf{F}) = \overline{\mathrm{W}}(\mathbf{F}) \,. \tag{2.23}$$

Choosing $\mathbf{R} = \sqrt{\mathbf{C}}\mathbf{F}^{-1}$ it is possible to rewrite the equation (2.23) as:

$$\overline{\mathrm{W}}(\mathbf{F}) = \overline{\mathrm{W}}(\sqrt{\mathbf{C}}) = \widetilde{\mathrm{W}}(\mathbf{C}) \,. \tag{2.24}$$

Hence the axiom of frame indifference implies that the strain-energy function depends only on the right Cauchy-Green tensor $\mathbf{C}$. Using equation (2.22) and the definition of $\mathbf{F}$ we have:

$$\mathrm{d}\overline{\mathrm{W}} = \frac{\partial \widetilde{\mathrm{W}}(\mathbf{C})}{\partial \mathbf{C}} : \mathrm{d}\mathbf{C} \,. \tag{2.25}$$

Finally using the definition (2.3), the simmetry of the right Cauchy-Green tensor and using the chain rule:

$$\mathrm{d}\overline{\mathrm{W}} = \frac{\partial\widetilde{\mathrm{W}}}{\partial\mathbf{C}} : \mathrm{d}\mathbf{C} = 2\mathbf{F}\frac{\partial\widetilde{\mathrm{W}}}{\partial\mathbf{C}} : \mathrm{d}\mathbf{F}\,. \tag{2.26}$$

Therefore, the first Piola-Kirchhoff stress tensor assumes the following form:

$$\mathbf{P} = 2\mathbf{F}\frac{\partial\widetilde{\mathrm{W}}}{\partial\mathbf{C}}\,. \tag{2.27}$$

By inversion of the relation (2.6) it is possible to define also the Cauchy stress tensor $\mathbf{T}_\mathrm{s}$ that reads:

$$\mathbf{T}_\mathrm{s} = 2\mathrm{J}^{-1}\mathbf{F}\frac{\partial\widetilde{\mathrm{W}}}{\partial\mathbf{C}}\mathbf{F}^\mathrm{T}\,. \tag{2.28}$$

To rewrite the strain-energy function for isotropic materials it is possible to use the *Rivlin-Ericksen* representation theorem:

Theorem 2.1.1. *For any isotropic hyperelastic materials the strain-energy function can be written as:*

$$\widetilde{\mathrm{W}}(\widehat{\mathbf{x}}, \mathbf{F}) = \mathrm{W}(\widehat{\mathbf{x}}, \mathrm{I}_1(\mathbf{C}), \mathrm{I}_2(\mathbf{C}), \mathrm{I}_3(\mathbf{C}))\,. \tag{2.29}$$

For further information and proof of the theorem see also [26]. Hence for isotropic hyperelastic materials the expression that defines the stress tensors ((2.27) and (2.28)) can be rewritten only as a function of first derivative of the principal invariants of the right Cauchy-Green tensor $\mathbf{C}$. Employing the notation $\mathrm{I}_j = \mathrm{I}_j(\mathbf{C})$ the first Piola-Kirchhoff and Cauchy stress tensors becomes:

$$\begin{aligned}
\mathbf{P} &= 2\mathbf{F}\left[\Big(\frac{\partial\mathrm{W}}{\partial\mathrm{I}_1} + \mathrm{I}_1\frac{\partial\mathrm{W}}{\partial\mathrm{I}_2}\Big)\mathbf{I} - \frac{\partial\mathrm{W}}{\partial\mathrm{I}_2}\mathbf{C} + \mathrm{I}_3\frac{\partial\mathrm{W}}{\partial\mathrm{I}_3}\mathbf{C}^{-1}\right];\\
\mathbf{T}_\mathrm{s} &= 2\mathrm{I}_3^{-1/2}\left[\Big(\frac{\partial\mathrm{W}}{\partial I_1} + \mathrm{I}_1\frac{\partial\mathrm{W}}{\partial\mathrm{I}_2}\Big)\mathbf{B} - \frac{\partial\mathrm{W}}{\partial\mathrm{I}_2}\mathbf{B}^2 + \mathrm{I}_3\frac{\partial\mathrm{W}}{\partial\mathrm{I}_3}\mathbf{I}\right].
\end{aligned} \tag{2.30}$$

Where $\mathbf{B}$ is the left Cauchy-Green tensor defined as $\mathbf{B} = \mathbf{F}\mathbf{F}^\mathrm{T}$.

Remark 2. Any compressible hyperelastic material depends on volume changes through the third principal invariant I_3. In fact I_3 is related to J by the following relation: $\mathrm{I}_3 = \mathrm{J}^2$.

Hyperelastic incompressible materials

Incompressible materials have to satisfy the constraint $\mathrm{J} = 1$. To be more precise an incompressible material does not change volume during the deformation. An admissible deformation for this kind of materials is:

$$\begin{cases} \mathcal{L}(\widehat{\mathbf{x}}, \mathrm{t}) = \mathbf{F}(\mathrm{t})\widehat{\mathbf{x}} + \mathbf{c}(\mathrm{t})\,, \\ \det(\mathbf{F}) = 1\,. \end{cases} \tag{2.31}$$

It is possible to prove [26] that the first Piola-Kirchhoff tensor has the following expression:

$$\mathbf{P} = \frac{\partial \overline{\mathrm{W}}}{\partial \mathbf{F}}(\mathbf{F}) - p\frac{\partial \mathrm{J}}{\partial \mathbf{F}}\,, \tag{2.32}$$

where p is a scalar field called *pressure*. As for compressible materials, the axiom of frame indifference implies that $\overline{\mathrm{W}}$ is a funtion of right Cauchy-Green tensor $\mathbf{C}$ only. So, it is possible to derive the following relations for the stress tensors:

$$\mathbf{P} = 2\mathbf{F}\frac{\partial \widetilde{\mathrm{W}}}{\partial \mathbf{C}} - p\mathbf{F}^{-\mathrm{T}}\,, \mathbf{T}_{\mathrm{s}} = 2\mathbf{F}\frac{\partial \widetilde{\mathrm{W}}}{\partial \mathbf{C}}\mathbf{F}^{\mathrm{T}} - p\mathbf{I}\,. \tag{2.33}$$

Using the same considerations made for compressible materials it is possible to define the stress tensors as a function of the invariants of $\mathbf{C}$:

$$\mathbf{P} = 2\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{I}_1} + \mathrm{I}_1\frac{\partial \mathrm{W}}{\partial \mathrm{I}_2}\Big)\,\mathbf{F} - 2\frac{\partial \mathrm{W}}{\partial \mathrm{I}_2}\mathbf{F}\mathbf{C} - p\mathbf{F}^{-\mathrm{T}}\,. \tag{2.34}$$

For incompressible materials the hydrostatic pressure plays the role of Lagrange multiplier associated to the incompressibility constraint $\mathrm{J} = 1$.

Hyperelastic nearly incompressible materials

A nearly incompressible material can be associated to any incompressible material. In fact, it is possible to demonstrate that for any strain energy function of an incompressible material $\mathrm{W}_{\mathrm{inc}}$ there exists a nearly incompressible material with the following constitutive law:

$$\mathbf{P} = \frac{\partial \mathrm{W}_{\epsilon}}{\partial \mathbf{F}}(\widehat{\mathbf{x}}, \mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\eta})\,, \tag{2.35}$$

where

$$\mathrm{W}_{\epsilon} = \mathrm{W}_{\mathrm{inc}}(\widehat{\mathbf{x}}, [\det(\mathbf{F})]^{-1/3}\mathbf{F}) + \frac{1}{2\epsilon}(\det\mathbf{F} - 1)^2\,.$$

For practical applications it is common to divide the strain-energy function into two parts. The first one is the isochoric part $\mathrm{W}_{\mathrm{iso}}$ that preserving volume during deformation. The second one is the volumetric part $\mathrm{W}_{\mathrm{vol}}$ that depends on the Jacobian of the deformation gradient J:

$$\mathrm{W}(\mathrm{I}_1(\overline{\mathbf{C}}), \mathrm{I}_2(\overline{\mathbf{C}}), \mathrm{J}) = \mathrm{W}_{\mathrm{iso}}(\mathrm{I}_1(\overline{\mathbf{C}}), \mathrm{I}_2(\overline{\mathbf{C}})) + \mathrm{W}_{\mathrm{vol}}(\mathrm{J})\,, \tag{2.36}$$

where $\overline{\mathbf{C}}$ is the so-called unimodular right Cauchy-Green tensor, defined as $\overline{\mathbf{C}} = \mathrm{J}^{-2/3}\mathbf{C}$ and such that $\det(\overline{\mathbf{C}}) = 1$. The nearly incompressible materials are preferred to the incompressible materials because it is more common to have small compressibility in the hyperelastic materials and tipically also in the biological tissues [5]. Thus, the use of nearly incompressible materials is the correct choice to well describe the arterial wall. Moreover, the problem is simpler than in incompressible materials since we do not have to treat the incompressibility constraint. Nevertheless, the price to pay for the simplicity of the formulation is that the resultating problem is badly conditioned.

2.1.4 Well-posedness of the mathematical problem: Polyconvexity

It is not possible to choose the form of the strain-energy function W arbitrarily otherwise under some conditions it is not guaranteed the well-posedness of the mathematical problem:

$$\begin{cases} \widehat{\rho}\,\ddot{\boldsymbol{\eta}} & = \widehat{\rho}\,\mathbf{b} + \widehat{\nabla}\cdot\mathbf{P}\,, \\ \boldsymbol{\eta}(\widehat{\mathbf{x}},0) & = \boldsymbol{\eta}_0(\widehat{\mathbf{x}})\,, \\ \dot{\boldsymbol{\eta}}(\widehat{\mathbf{x}},0) & = \dot{\boldsymbol{\eta}}_0(\widehat{\mathbf{x}})\,, \\ \boldsymbol{\eta}(\widehat{\mathbf{x}},\mathrm{t}) & = \boldsymbol{g}\,, \qquad \widehat{\mathbf{x}}\in\widehat{\Gamma}_D\,, \\ \mathbf{P}\widehat{\mathbf{n}}(\widehat{\mathbf{x}},\mathrm{t}) & = \boldsymbol{h}\,, \qquad \widehat{\mathbf{x}}\in\widehat{\Gamma}_N\,, \end{cases} \tag{2.37}$$

To have the correct solution of (2.37) it is necessary that W respects some constitutive inequalities. The most important, in this sense, is the Baker-Ericksen inequality:

$$(t_i - t_j)(\lambda_i - \lambda_j) \geq 0 \quad 1 \leq i \neq j \leq 3\,, \tag{2.38}$$

where t_i, t_j are the principal stresses (eigenvalues of the Cauchy stress tensor) and λ_i, λ_j are the corresponding eigenvalues of $\mathbf{F}$ [28].
To satisfy (2.38), it is sufficient the strict ellipticity of the strain-energy function for all deformations, that is W must satisfy the following *strict Legendre-Hadamard condition*:

Theorem 2.1.2 (**Legendre-Hadamard**). *The strain-energy function* $\mathrm{W}\in\mathrm{C}^2(\mathbb{M}^{3\times3},\mathbb{R})$ *is strictly elliptic if and only if:*

$$\begin{aligned} &\forall\mathbf{F}\in\mathbb{M}^{3\times3},\ \ \forall\boldsymbol{\upsilon},\boldsymbol{\varepsilon}\in\mathbb{R}^3: \\ &\frac{\mathrm{D}^2W(\mathbf{F})}{\mathrm{D}\mathbf{F}^2}\cdot(\boldsymbol{\upsilon}\otimes\boldsymbol{\varepsilon},\,\boldsymbol{\upsilon}\otimes\boldsymbol{\varepsilon})\ >\ 0\,. \end{aligned} \tag{2.39}$$

Another method can be employed to solve problem (2.14): the minimization of the strain-energy function with respect to the deformations. It is possible to demonstrate that the well-posedness, is ensured by the *quasi-convexity* of the strain-energy function [29]. This request is very difficult to verify. A simpler sufficient condition is the request of *polyconvexity* [30].

Definition 2.1. Polyconvexity: Let $\mathrm{W}\in\mathrm{C}^2(\mathbb{M}^{3\times3},\mathbb{R})$, be a given scalar-valued energy density. The function $\mathbf{F}\to\mathrm{W}(\mathbf{F})$ is polyconvex if and only if there exists a function G: $\mathbb{M}^{3\times3}\times\mathbb{M}^{3\times3}\times\mathbb{R}\to\mathbb{R}$ such that:

$$\mathrm{W}(\mathbf{F}) = \mathrm{G}(\mathbf{F},\mathrm{Adj}\mathbf{F},\mathrm{det}\mathbf{F})\,, \tag{2.40}$$

where $\mathrm{Adj}\mathbf{F} = (\mathrm{Cof}\mathbf{F})^{\mathrm{T}}$ is the *Adjugate* of $\mathbf{F}$, and the function $\mathrm{G}(\cdot,\cdot,\cdot)$ is convex.

From this point of view, it is very simple to verify that the following strain-energy function

$$\mathrm{W}(\mathbf{F}) = \mathrm{W}_1(\mathbf{F}) + \mathrm{W}_2(\mathrm{Adj}\mathbf{F}) + \mathrm{W}_3(\mathrm{det}\mathbf{F}) \tag{2.41}$$

is polyconvex if and only if W_1, W_2, W_3 are convex with respect to their arguments.
The polyconvexity property guarantees two main things. The first one is the existence of

solutions to all boundary value problem with physical boundary conditions. The second one is the ellipticity of W for all deformations. Hence, if the strain-energy function is polyconvex, the mathematical problem (2.14) is well-posed.

In the next paragraph we introduce the structural models we have implemented and we specify if the strain-energy function satisfies the constraint of polyconvexity.

2.1.5 Structural models implemented

St.Venant-Kirchhoff model

The St.Venant-Kirchhoff model is the simplest compressible material. Its strain-energy function is defined as a quadratic isotropic function of the Green-Lagrange tensor:

$$\mathrm{W} = \mathrm{W}(\mathbf{E}) = \frac{\lambda}{2}(\mathrm{tr}\mathbf{E})^2 + \mu \mathrm{tr}(\mathbf{E}^2)\,, \tag{2.42}$$

where λ and μ are the first and the second Lamè constants of the material:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}\,, \tag{2.43}$$

$$\mu = \frac{E}{2(1+\nu)}\,, \tag{2.44}$$

where ν and E are respectively the poisson's ratio and the Young modulus. Using first equation of (2.30) it is possible to define the corrisponding first Piola-Kirchhoff stress tensor in terms of $\mathbf{F}$ and $\mathbf{E}$:

$$\mathbf{P} = \frac{\lambda}{2}(\mathrm{I}_1(\mathbf{C}) - 3)\mathbf{F} - \mu\mathbf{F} + \mu\mathbf{F}\mathbf{C}\,. \tag{2.45}$$

It is more convenient to rewrite (2.45) in terms of displacements $\boldsymbol{\eta}$:

$$\begin{aligned}\mathbf{P}(\boldsymbol{\eta}) \;&= \lambda(\boldsymbol{\nabla}\cdot\boldsymbol{\eta})\mathbf{I} + \mu(\boldsymbol{\nabla}\boldsymbol{\eta} + \boldsymbol{\nabla}\boldsymbol{\eta}^{\mathrm{T}}) \\ &+\frac{\lambda}{2}(\boldsymbol{\nabla}\boldsymbol{\eta} : \boldsymbol{\nabla}\boldsymbol{\eta})\mathbf{I} + \mu\boldsymbol{\nabla}\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{\nabla}\boldsymbol{\eta} \\ &+\lambda(\boldsymbol{\nabla}\cdot\boldsymbol{\eta})\boldsymbol{\nabla}\boldsymbol{\eta} + \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} \\ &+\mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}^{\mathrm{T}}) + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}\,.\end{aligned} \tag{2.46}$$

This material does not satisfies the constraint of policonvexity. Because of this, it is not usually used for problems with large levels of deformation. More details can be found in [31]-[27].

Neo-Hookean nearly incompressible model

The Neo-Hookean is a model of a nearly incompressible material. It presents a behaviour similar to a rubber. We are interested in this model because it can represent the first part of

the stress-strain curve (first stages of deformation), using a multi-mechanism model, introduced in Chapter 1. The strain-energy function of Neo-Hookean material is divided into two parts: the isochoric part and the volumetric part:

$$\mathrm{W} = \frac{\mu}{2}\big(\mathrm{I}_1(\overline{\mathbf{C}}) - 3\big) + \frac{\kappa}{4}\big[(\mathrm{J}-1)^2 + (\ln \mathrm{J})^2\big]\,. \tag{2.47}$$

The first Piola-Kirchhoff stress tensor is also divided into isochoric and volumetric part. After some calculations it assumes the following form:

$$\mathbf{P} = \mu \mathrm{J}^{-2/3}\Big(\mathbf{F} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}^{-\mathrm{T}}\Big) + \mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}^{-\mathrm{T}}\,, \tag{2.48}$$

where κ is the bulk modulus of the material and has the dimensions of a pressure. It represents also the penalty to the volumetric part with respect the isochoric part. In particular, for high value of κ we have incompressible materials. For Neo-Hookean nearly incompressible material the polyconvexity of the strain-energy function W is satisfied. Hence the mathematical problem is well-posed.

Exponential nearly incompressible model

As the Neo-hookean model, the exponential model used is a nearly incompressible material. In particular, this material is widely used for soft bilogical tissues like the walls of the arteries. From a multi-mechanism point-of-view, Exponential model can be used as a second part of the curve, for large deformations. The strain-energy function, also in this case, is divided into isochoric and volumetric part:

$$\mathrm{W} = \frac{\alpha}{2\gamma}\big(e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)} - 1\big) + \frac{\kappa}{4}\big[(\mathrm{J}-1)^2 + (\ln \mathrm{J})^2\big]\,, \tag{2.49}$$

where α and γ are respectively the pre-exponential coefficient and the exponetial coefficient. The first Piola-Kirchhoff stress tensor after some calculation reads:

$$\mathbf{P} = \alpha \mathrm{J}^{-2/3}\Big(\mathbf{F} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}^{-\mathrm{T}}\Big)e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)} + \mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}^{-\mathrm{T}}\,. \tag{2.50}$$

Also in this case the mathematical problem is well-posed, because the strain-energy function W satifies the policonvexity constraint.

Remark 3. The same volumetric part is used for both nearly incompressible models. It is possible to relate the bulk modulus κ with the poisson's ratio and the Young modulus:

$$\kappa = \frac{\mathrm{E}}{3(1-2\nu)}\,, \tag{2.51}$$

for the consistency of the nonlinear consitutive laws with the linear constitutive law, in the region of small deformations. In the structural and FSI numerical analysis these coefficients

are not strictly related with (2.51), but we have used typical values for biological tissues from the literature. In fact, [26], suggest to use a parameter κ in the following range:

$$\mu 10^2 \leq \kappa \leq \mu 10^6$$

This choice does not influence strongly the comparison between materials because the stress-strain response in the region of small deformations is very similar for nonlinear structural models and linear elasticity also with our parameters.
However, for a systematic comparative analysis between the different consitutive laws, could be necessary to use the relation (2.51) or, eventually, another one that takes into account a possible linearization with a pre-stress model.

2.2 Finite-element formulation

The finite-element formulation is the core of the solver. Here the matrix of the system are computed and the overall algebraic system are assembled. The solution of a time-varying problem needs also of a temporal scheme to integrate the equations. In this section we give a brief explanation of these issues.

2.2.1 Weak formulation of the structural problem

We derive the weak form in a formal way by multiplying the equation (2.14) by a test function $\delta\mathbf{v}$ and integrating on $\widehat{\Omega}$.

$$\int_{\widehat{\Omega}} \widehat{\rho}\,\ddot{\boldsymbol{\eta}} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega} = \int_{\widehat{\Omega}} \widehat{\rho}\,\mathbf{b} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega} + \int_{\widehat{\Omega}} (\widehat{\nabla} \cdot \mathbf{P}) \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega}\,. \tag{2.52}$$

In particular the test function has to respect the constraint:

$$\delta\mathbf{v} = \mathbf{0} \quad \text{on } \widehat{\Gamma}_{\mathrm{D}} \tag{2.53}$$

Using the Gauss theorem and the boundary conditions of (2.16) we obtain:

$$\int_{\widehat{\Omega}} \widehat{\rho}\,\ddot{\boldsymbol{\eta}} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega} = \int_{\widehat{\Omega}} \widehat{\rho}\,\mathbf{b} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega} - \int_{\widehat{\Omega}} \widehat{\nabla}(\delta\mathbf{v}) : \mathbf{P}\,\mathrm{d}\widehat{\Omega} + \int_{\widehat{\partial\Omega}} \widehat{\mathbf{t}} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\partial\Omega}\,. \tag{2.54}$$

If we assume that $\delta\mathbf{v}$ is a velocity, so, $\widehat{\nabla}(\delta\mathbf{v})$ is equal to $\delta\dot{\mathbf{F}}$ and the last equation assumes the following expression:

$$\int_{\widehat{\Omega}} \widehat{\rho}\,\ddot{\boldsymbol{\eta}} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega} = \int_{\widehat{\Omega}} \widehat{\rho}\,\mathbf{b} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\Omega} - \int_{\widehat{\Omega}} \delta\dot{\mathbf{F}} : \mathbf{P}\,\mathrm{d}\widehat{\Omega} + \int_{\widehat{\partial\Omega}} \widehat{\mathbf{t}} \cdot \delta\mathbf{v}\,\mathrm{d}\widehat{\partial\Omega}\,. \tag{2.55}$$

In particular, we are interested on the stiffness term $\int_{\widehat{\Omega}} \delta\dot{\mathbf{F}} : \mathbf{P}\,\mathrm{d}\widehat{\Omega}$ that depends by the constitutive law adopted.

We are now in the position of giving the correct functional setting of (2.14) by employing the Sobolev spaces introduced in appendix A.

Weak-formulation 2.1 (Continuous setting)**.** *For any $t > 0$ find $\boldsymbol{\eta} = \boldsymbol{\eta}(t) \in \mathrm{V}(\widehat{\Omega})$:*

$$\begin{cases} \displaystyle\int_{\widehat{\Omega}} \widehat{\rho}\, \ddot{\boldsymbol{\eta}} \cdot \boldsymbol{\phi}\, d\widehat{\Omega} + a(\boldsymbol{\eta}, \boldsymbol{\phi}) = F(\boldsymbol{\phi}) \qquad \forall \boldsymbol{\phi} \in \mathrm{V}(\widehat{\Omega})\,, \\ \boldsymbol{\eta}(0) = \boldsymbol{\eta}_0\,, \\ \dot{\boldsymbol{\eta}}(0) = \dot{\boldsymbol{\eta}}_0\,, \end{cases} \tag{2.56}$$

where $\mathrm{V}(\widehat{\Omega}) = \mathrm{H}^1_0(\widehat{\Omega})$ and bilinear form $a(\cdot,\cdot)$, and right-hand side $F(\cdot)$ assume the following definitions:

$$\begin{aligned} a(\boldsymbol{\eta}, \boldsymbol{\phi}) &= \int_{\widehat{\Omega}} \mathbf{P} : \widehat{\boldsymbol{\nabla}} \boldsymbol{\phi}\, d\widehat{\Omega}\,, \\ F(\boldsymbol{\phi}) &= \int_{\partial\widehat{\Omega}} \widehat{\mathbf{t}} \cdot \boldsymbol{\phi}\, d\widehat{\partial\Omega} + \int_{\widehat{\Omega}} \widehat{\rho}\, \mathbf{b} \cdot \boldsymbol{\phi}\, d\widehat{\Omega}\,. \end{aligned} \tag{2.57}$$

We have assumed, for simplicity, only homogeneous Dirichlet boundary conditions (for non-homogeneous Dirichlet boundary conditions see for example [32]. The discrete version of (2.56) is obtained by the Galerkin method [32] where $\mathrm{V}(\widehat{\Omega})$ is replaced by a subspace V_h of finite dimension. As usual a finite element formulation employs a mesh of the reference domain $\widehat{\Omega}$ to build V_h. The space discretization is thus defined as:

$$\widehat{\Omega} = \bigcup_{\mathrm{K} \in \widehat{\tau}_\mathrm{h}} \mathrm{K}\,, \tag{2.58}$$

where K indicates the generic element of the mesh $\widehat{\tau}_\mathrm{h}$. Using Lagrangian finite elements, we have the following functional space:

$$\chi^r_\mathrm{h} = \{\phi_\mathrm{h} \in \mathrm{C}^0(\widehat{\Omega}_\mathrm{h}) : \phi_\mathrm{h}\,|_{K_j} \in \mathbb{P}_r\,,\ \forall K_j \in \widehat{\tau}_\mathrm{h}\,,\} \qquad r = 1, 2, \dots\,, \tag{2.59}$$

where $\mathbb{P}_r$ is the polynomial space of degree r. V_h is then a subspace of χ^r_h obtained by imposing the essential boundary conditions. The discrete version of (2.56) becomes:

Weak-formulation 2.2 (Discrete setting)**.** *For any $t > 0$ find $\boldsymbol{\eta}_h = \boldsymbol{\eta}_h(t) \in V_h(\widehat{\Omega}_h)$:*

$$\begin{cases} \displaystyle\int_{\widehat{\Omega}_h} \widehat{\rho}\, \ddot{\boldsymbol{\eta}}_h \cdot \boldsymbol{\phi}_h\, d\widehat{\Omega} + a(\boldsymbol{\eta}_h, \boldsymbol{\phi}_h) = F(\boldsymbol{\phi}_h) \qquad \forall \boldsymbol{\phi}_h \in V_h(\widehat{\Omega}_h) \\ \boldsymbol{\eta}_h(0) = \boldsymbol{\eta}_{h_0} \\ \dot{\boldsymbol{\eta}}_h(0) = \dot{\boldsymbol{\eta}}_{h_0} \end{cases} \tag{2.60}$$

Finally, introducing the following approximate solution, associated to the discretization:

$$\boldsymbol{\eta}_\mathrm{h}(\widehat{\mathbf{x}}) = \sum_{j \in \mathrm{N}_\mathrm{h}} \boldsymbol{\eta}_j \boldsymbol{\phi}_j(\widehat{\mathbf{x}})\,, \tag{2.61}$$

where the ϕ_j are a basis of $\mathrm{V_h}$, it is possible to rewrite the discrete weak formulation as a system of ordinary differential equations (ODEs):

$$\mathbf{M}\ddot{\boldsymbol{\eta}} + \mathbf{k}(\boldsymbol{\eta}) = \mathbf{f}\,, \tag{2.62}$$

where $\boldsymbol{\eta}$ is the vector of the unknowns of the discret problem and the other components are defined as follow:

$$\begin{aligned} \mathrm{M}_{ij} &= \int_{\widehat{\Omega}_\mathrm{h}} \widehat{\rho}\,\boldsymbol{\phi}_j\boldsymbol{\phi}_i \,\mathrm{d}\widehat{\Omega}\,, \\ k(\boldsymbol{\eta})_i &= \int_{\widehat{\Omega}_\mathrm{h}} \mathbf{P}\big(\sum_{j\in\mathrm{N_h}} \boldsymbol{\eta}_j\boldsymbol{\phi}_j(\widehat{\mathbf{x}})\big) : \boldsymbol{\nabla}\boldsymbol{\phi}_i \,\mathrm{d}\widehat{\Omega}\,, \\ f_i &= \int_{\widehat{\Omega}_\mathrm{h}} \sum_{j\in N_\mathrm{h}} t_{0_j}\phi_j\phi_i \,\mathrm{d}\widehat{\Omega} + \int_{\widehat{\Omega}_\mathrm{h}} \sum_{j\in N_\mathrm{h}} b_j\phi_j\phi_i \,\mathrm{d}\widehat{\Omega}\,. \end{aligned} \tag{2.63}$$

In particular M_{ij} is the mass matrix, $k(\boldsymbol{\eta})_i$ is the non-linear stiffness vector and f_i is the known term.

2.2.2 Time discretization

The Newmark scheme is used to perform the time-discretization. In particular, from equation (2.62), the time discretization is obtained in the following manner:

- Definition of a time interval $\mathbb{I} = [0, \mathrm{T}]$;
- Uniform discretization of interval $\mathbb{I} \to \mathbb{I}_n = [\mathrm{t}^n, \mathrm{t}^{n+1}]$, with $\delta\mathrm{t} = \mathrm{t}^{n+1} - \mathrm{t}^n$;
- Application of the Newmark scheme to system of ODEs (2.62)

$$\begin{cases} \mathbf{M}\ddot{\boldsymbol{\eta}}^{n+1} + \mathbf{k}(\boldsymbol{\eta}^{n+1}) = \mathbf{f}^{n+1}\,, \\ \dot{\boldsymbol{\eta}}^{n+1} = \dot{\boldsymbol{\eta}}^n + \delta\mathrm{t}[(1-\theta)\ddot{\boldsymbol{\eta}}^n + \theta\ddot{\boldsymbol{\eta}}^{n+1}]\,, \\ \ddot{\boldsymbol{\eta}}^{n+1} = \dfrac{2}{\zeta\delta\mathrm{t}^2}\boldsymbol{\eta}^{n+1} - \dfrac{2}{\zeta\delta\mathrm{t}^2}(\boldsymbol{\eta}^n + \delta\mathrm{t}\dot{\boldsymbol{\eta}}^n) - \dfrac{1-\zeta}{\zeta}\ddot{\boldsymbol{\eta}}^n\,, \end{cases} \tag{2.64}$$

$$\downarrow$$

$$\boxed{\begin{aligned} \frac{2}{\delta\mathrm{t}^2}\mathbf{M}\boldsymbol{\eta}^{n+1} + \zeta\mathbf{k}(\boldsymbol{\eta}^{n+1}) \;&=\; \frac{2}{\delta\mathrm{t}^2}\mathbf{M}(\boldsymbol{\eta}^n + \delta\mathrm{t}\dot{\boldsymbol{\eta}}^n) - (1-\zeta)\mathbf{k}(\boldsymbol{\eta}^n) \\ &\quad +(1-\zeta)\mathbf{f}^n + \zeta\mathbf{f}^{n+1}\,, \end{aligned}} \tag{2.65}$$

where we have set: $\mathrm{g}^s = \mathrm{g}(\mathrm{t}^s)$. Here, ζ and θ are the coefficients of the Newmark's method. The stability of the Newmark's method depends on the choice of ζ and θ and from the viscous damping (when it is present). The following relations describe the stability conditions for a general Newmark's method applied to a second order dynamical system without viscous damping (see also [33]).

$$\boxed{\textit{Unconditional stability} \qquad \zeta \geq \theta \geq \tfrac{1}{2}\,,} \tag{2.66}$$

$$\boxed{\textit{Conditional stability} \qquad \begin{aligned} \theta &\geq \tfrac{1}{2}\,, \\ \zeta &< \theta\,, \\ \omega^h \delta\mathrm{t} &\leq \Omega_{crit}\,, \end{aligned}} \tag{2.67}$$

where ω^h is the maximum natural frequency and Ω_{crit} is defined by the following relation:

$$\Omega_{crit} = \left(\frac{\theta}{2} - \frac{\zeta}{2}\right)^{-\frac{1}{2}}. \tag{2.68}$$

In the table 2.1 we summarize the well-known members of the Newmark family of methods [33]:

Table 2.1: Well-known Newmark methods

Method	Type	ζ	θ	Stability condition	OA
Average acceleration	Implicit	0.5	0.5	*Unconditional*	2
Linear acceleration	Implicit	0.33	0.5	*Conditional*	2
Fox-Goodwin	Implicit	0.1667	0.5	*Conditional*	2
Central difference	Explicit	0	0.5	*Conditional*	2

The choice of Average acceleration method gives the optimal time convergence rate. However it may cause spurios oscillations since there is no numerical damping. It is possible to introduce damping by using a parameter ζ bigger than 0.5. For example the choice of $\zeta = 1$ and $\theta = 0.5$ is a fully dissipative method that cuts the spurious oscillations of the solution. This choice has been used for quasi-static problems. For dynamical problems it is necessary to have a good order of accurancy (OA), hence a second order method like average acceleration is preferred.

2.2.3 Linearization

As previously remarked, the stiffness term $\mathbf{k}(\boldsymbol{\eta})$ is non-linear with respect to the displacement $\boldsymbol{\eta}$. Hence to solve the system (2.62) it is necessary to find the solution as the limit of solutions of suitable linearized problems. The linearization is obtained by a Newton method. In fact, the solution of (2.62) is equivalent to find the root $\boldsymbol{\eta}$ of the following problem:

$$\boxed{\begin{array}{l}\mathcal{Z}(\boldsymbol{\eta}) = \dfrac{2}{\delta \mathrm{t}^2}\mathbf{M}\boldsymbol{\eta} + \zeta\mathbf{k}(\boldsymbol{\eta}) - \dfrac{2}{\delta \mathrm{t}^2}\mathbf{M}(\boldsymbol{\eta}^n - \delta \mathrm{t}\dot{\boldsymbol{\eta}}^n) + (1-\zeta)\mathbf{k}(\boldsymbol{\eta}^n) + \\ -(1-\zeta)\mathbf{f}^n - \zeta\mathbf{f}^{n+1} = 0\,.\end{array}} \tag{2.69}$$

The unknown of the problem at each Newton iteration k is the displacement $\boldsymbol{\eta}^{(k)}$ which is set initially to $\boldsymbol{\eta}^n$. It is necessary to introduce the Jacobian[3] $\mathrm{J}_{\mathcal{Z}}(\boldsymbol{\eta}^{(k)}) = \mathrm{D}\mathcal{Z}(\boldsymbol{\eta}^{(k)})$ and the increment of solution $\boldsymbol{\delta\eta}^{(k)} = \boldsymbol{\eta}^{(k+1)} - \boldsymbol{\eta}^{(k)}$.
Hence the problem becomes:

Let $\boldsymbol{\eta}^{(0)} \in \mathbb{R}^3$ be given (for instance from the previous time-step), iterate for $k = 1, 2, ...$ until convergence:

$$\boxed{\begin{array}{ll}\text{solve} & \mathrm{J}_{\mathcal{Z}}(\boldsymbol{\eta}^{(k)})\boldsymbol{\delta\eta}^{(k)} = -\mathcal{Z}(\boldsymbol{\eta}^{(k)})\,, \\ \text{define} & \boldsymbol{\eta}^{(k+1)} = \boldsymbol{\eta}^{(k)} + \boldsymbol{\delta\eta}^{(k)}\,,\end{array}} \tag{2.70}$$

and set $\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^{k+1}$. The test for convergence is the infinity norm of the residual. In particular, defining a tolerance $\varepsilon_R = \overline{\varepsilon_{R_{abs}}} + \mid \mathcal{Z}(\boldsymbol{\eta}^{(0)}) \mid \overline{\varepsilon_{R_{rel}}}$ the stopping criterion for the Newton method is the following:

$$\| \mathcal{Z}(\boldsymbol{\eta}^{(k)}) \|_{\mathrm{L}^\infty(\widehat{\Omega})} < \varepsilon_R\,. \tag{2.71}$$

Remark 4. The test for convergence of the Newton method is in fact imposed on the forces.

To solve the linearized system (2.70) the preconditioned GMRES (P-GMRES) method [34] has been used. The criterion to stop the P-GMRES method is to compare the norm of displacement $\boldsymbol{\eta}$ between iteration $\ell + 1$ and iteration ℓ of the method[4]. If $\varepsilon_{P-GMRES}$ is the tolerance of P-GMRES, the convergence test is:

$$\| \boldsymbol{\eta}_{(\ell+1)} - \boldsymbol{\eta}_{(\ell)} \|_{\mathrm{L}^\infty(\widehat{\Omega})} < \varepsilon_{P-GMRES}\,. \tag{2.72}$$

When P-GMRES method reaches to convergence, the solution is used to recompute the system (2.70) and in particular the residual of the system to verify the convergence of Newton method. The P-GMRES method will be more efficient when the number of iterations for convergence is small. Summarizing, the steps for the solution of the nonlinear system (2.62) at each time-step are the following:

1. Choose an initial guess $\boldsymbol{\eta}^{(0)} \in \mathbb{R}^3$ and choose the tolerances $\overline{\varepsilon_R}$ and ε_{GMRES};
2. Build the system (2.62), compute the Jacobian of the system and the residual;
3. Enter into Newton's loop and compare the residual with the tolerance ε_R;
4. If the residual is larger than ε_R, solve the linearized system (2.70) with P-GMRES method until $\| \boldsymbol{\eta}_{(\ell+1)} - \boldsymbol{\eta}_{(\ell)} \|_{\mathrm{L}^\infty(\widehat{\Omega})} < \varepsilon_{P-GMRES}$;

[3] $\mathrm{D}\mathcal{Z}(\boldsymbol{\eta}^{(k)})$ is the directional derivative of $\mathcal{Z}(\boldsymbol{\eta}^{(k)})$. For more details see [9], [31]
[4] ℓ is the iteration of the P-GMRES method

5. Re-build the system (2.62) (step 2);
6. Compare the residual with the tolerance of Newton method ε_R;
7. If the residual is larger than ε_R return to step 4, else the solution has been found and it is possible to go to the next time-step;

2.2.4 Quadrature rules and computation of integrals

The calculation of the integrals required in the finite element formulation of the structural problem, are obtained by quadrature formulas. For each quadrature point a local tensor is defined. A loop on quadrature points perform the computation of the integrals. In particular for a generic stiffness term, we have:

$$\int_{\mathrm{K}} \mathbf{P}(\boldsymbol{\eta}_{\mathrm{h}}) : \widehat{\boldsymbol{\nabla}}\phi_i \mathrm{dV}_{\mathrm{K}} \approx \sum_{ig} \mathbf{P}(\boldsymbol{\eta})_{ig} \widehat{\boldsymbol{\nabla}}\phi_{ig}\, \omega_{ig}\,, \tag{2.73}$$

where ig indicate the quadrature point and ω_{ig} are the corresponding weights. Here we summarize the terms that we have implemented in the classes *elemOper* and *elemOperStructure*. In particular, we associate for each stiffness term of each material the corresponding method.

Stiffness term computation: St.Venant-Kirchhoff

We are interested in the computation of the stiffness term (2.73). For St.Venant-Kirchhoff material, the first Piola-Kirchhoff tensor is defined in (2.46), here reported:

$$\begin{aligned}\mathbf{P}(\boldsymbol{\eta}) &= \lambda(\widehat{\boldsymbol{\nabla}}\cdot\boldsymbol{\eta})\mathbf{I} + \mu(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}^{\mathrm{T}}) \\ &+\frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})\mathbf{I} + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} \\ &+\lambda(\widehat{\boldsymbol{\nabla}}\cdot\boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} \\ &+\mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}^{\mathrm{T}}) + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}\,.\end{aligned}$$

The calculation of the bilinear form $\mathrm{a}(\boldsymbol{\eta}, \boldsymbol{\phi})$ is performed as:

$$\begin{aligned}\int_{\mathrm{K}} \mathbf{P}(\boldsymbol{\eta}_{\mathrm{h}}) : \widehat{\boldsymbol{\nabla}}\phi_i \mathrm{dV}_{\mathrm{K}} &\approx \textstyle\sum_{ig} \Big(\lambda(\widehat{\boldsymbol{\nabla}}\cdot\boldsymbol{\eta}_{ig})\mathbf{I} + \mu(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}}) \\ &+\frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig})\mathbf{I} + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} \\ &+\lambda(\widehat{\boldsymbol{\nabla}}\cdot\boldsymbol{\eta}_{ig})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} + \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} \\ &+\mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}}) + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\Big)\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}\,,\end{aligned} \tag{2.74}$$

where each term is defined in the following method into *elemOper* class:

1. $\sum_{ig} \lambda(\widehat{\boldsymbol{\nabla}}\cdot\boldsymbol{\eta}_{ig})\mathbf{I}\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_div;

2. $\sum_{ig} \mu(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}})\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_strain;

3. $\sum_{ig} \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig})\mathbf{I}\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_derdiv;

4. $\sum_{ig} \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_dergradbis;

5. $\sum_{ig} \lambda(\widehat{\boldsymbol{\nabla}} \cdot \boldsymbol{\eta}_{ig})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_divgrad;

6. $\sum_{ig} \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_gradgrad;

7. $\sum_{ig} \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}})\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_dergrad_gradbis;

8. $\sum_{ig} \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}_{ig}\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method stiff_gradgradTr_gradbis.

For St.Venant-Kirhhoff model, the linearization of the stiffness term is obtained using the directional derivatives [31]-[9]. Here, we write the linearization of the stiffness term for St.Venant-Kirchhoff model:

$$\begin{aligned}
\mathrm{D}\mathbf{P}(\boldsymbol{\eta})[\delta\boldsymbol{\eta}] \;&= \lambda(\widehat{\boldsymbol{\nabla}} \cdot \delta\boldsymbol{\eta}) + \mu(\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} + (\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta})) + \lambda\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} \\
&+\lambda(\widehat{\boldsymbol{\nabla}} \cdot \boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} + \lambda(\widehat{\boldsymbol{\nabla}} \cdot \delta\boldsymbol{\eta}) + \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} \\
&+\frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \frac{\lambda}{2}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} : \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} \\
&+\mu(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} + \mu(\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta})^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta} + \mu\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} \\
&+\mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta})^{\mathrm{T}} + \mu\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})^{\mathrm{T}} + \mu\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} \\
&+\mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta})^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \mu\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta})^{\mathrm{T}}\widehat{\boldsymbol{\nabla}}\delta\boldsymbol{\eta}
\end{aligned} \tag{2.75}$$

Stiffness term computation: Neo-Hookean

We are interested in the computation of the stiffness term (2.73). For St.Venant-Kirchhoff material, the first Piola-Kirchhoff tensor is defined in (2.48), here reported:

$$\mathbf{P} = \mu \mathrm{J}^{-2/3}\Big(\mathbf{F} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}^{-\mathrm{T}}\Big) + \mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}^{-\mathrm{T}}\,,$$

The calculation of the bilinear form $\mathrm{a}(\boldsymbol{\eta}, \boldsymbol{\phi})$ is performed as:

$$\begin{aligned}
\int_{\mathrm{K}} \mathbf{P}(\mathbf{F}_{\mathrm{h}}) : \widehat{\boldsymbol{\nabla}}\phi_i \mathrm{dV}_{\mathrm{K}} \;&\approx \sum_{ig} \Big[\mu \mathrm{J}^{-2/3}\Big(\mathbf{F}_{ig} - \tfrac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}_{ig}^{-\mathrm{T}}\Big) \\
&+\mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}_{ig}^{-\mathrm{T}}\Big]\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}\,,
\end{aligned} \tag{2.76}$$

where each term is defined in the following methods into *elemOperStructure* class:

1. $\sum_{ig}\Big[\mu \mathrm{J}^{-2/3}\Big(\mathbf{F}_{ig} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}_{ig}^{-\mathrm{T}}\Big]\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method source_P1iso_NH;

2. $\sum_{ig}\Big[\mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}^{-\mathrm{T}}\Big]\widehat{\boldsymbol{\nabla}}\phi_{ig}\,\omega_{ig}$: method source_Pvol;

The linearization of the stiffness term is performed with respect to $\mathbf{F}$. In particular, we introduce the fourth order tensor $\mathbb{C}$:

$$\mathbb{C} = \frac{\partial \mathbf{P}}{\partial \mathbf{F}}\,, \tag{2.77}$$

and we linearize the stiffness term using the directional derivatives with respect $\mathbf{F}$ separating the isochoric and the volumetric part:

$$\begin{aligned}\mathbb{C}_{\text{iso}} : \delta\mathbf{F} \;\; &= -\frac{2}{3}\mu \mathrm{J}^{-2/3}(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F} + \frac{2}{9}\mu \mathrm{I}_1(\overline{\mathbf{C}})(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}} \\ &\quad -\frac{2}{3}\mu \mathrm{J}^{2/3}(\mathbf{F} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}} + \mu \mathrm{J}^{-2/3}\delta\mathbf{F} + \frac{\mu}{3}\mathrm{I}_1(\overline{\mathbf{C}})\mathbf{F}^{-\mathrm{T}}\delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}\,.\end{aligned} \tag{2.78}$$

$$\begin{aligned}\mathbb{C}_{\text{vol}} : \delta\mathbf{F} \;\; &= \frac{\kappa}{2}\mathrm{J}\Big(2\mathrm{J} - 1 + \frac{1}{J}\Big)(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}} \\ &\quad -\frac{\kappa}{2}(\mathrm{J}^2 - \mathrm{J} + \ln\mathrm{J})\mathbf{F}^{-\mathrm{T}}\delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}\,.\end{aligned} \tag{2.79}$$

where each term is defined in the following methods into *elemOperStructure* class:

1. $-\frac{2}{3}\mu \mathrm{J}^{-2/3}(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}$: method stiff_Jac_P1iso_NH_1term;

2. $\frac{2}{9}\mu \mathrm{I}_1(\overline{\mathbf{C}})(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_P1iso_NH_2term;

3. $-\frac{2}{3}\mu \mathrm{J}^{2/3}(\mathbf{F} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_P1iso_NH_3term;

4. $\mu \mathrm{J}^{-2/3}\delta\mathbf{F}$: method stiff_Jac_P1iso_NH_4term;

5. $\frac{\mu}{3}\mathrm{I}_1(\overline{\mathbf{C}})\mathbf{F}^{-\mathrm{T}}\delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_P1iso_NH_5term;

6. $\frac{\kappa}{2}\mathrm{J}\Big(2\mathrm{J} - 1 + \frac{1}{J}\Big)(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_Pvol_1term;

7. $-\frac{\kappa}{2}(\mathrm{J}^2 - \mathrm{J} + \ln\mathrm{J})\mathbf{F}^{-\mathrm{T}}\delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_Pvol_2term;

Stiffness term computation: Exponential

We are interested in the computation of the stiffness term (2.73). For St.Venant-Kirchhoff material, the first Piola-Kirchhoff tensor is defined in (2.50), here reported:

$$\mathbf{P} = \alpha \mathrm{J}^{-2/3}\Big(\mathbf{F} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}^{-\mathrm{T}}\Big)e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)} + \mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}^{-\mathrm{T}}\,.$$

The calculation of the bilinear form $\mathrm{a}(\boldsymbol{\eta}, \boldsymbol{\phi})$ is performed as:

$$\begin{aligned}\int_{\mathrm{K}} \mathbf{P}(\mathbf{F}_{\mathrm{h}}) : \widehat{\boldsymbol{\nabla}}\phi_i \mathrm{dV}_{\mathrm{K}} \;\; &\approx \sum_{ig} \Big[\alpha \mathrm{J}^{-2/3}\Big(\mathbf{F}_{ig} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}_{ig}^{-\mathrm{T}}\Big) e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)} \\ &\quad + \mathrm{J}\frac{\kappa}{2}\Big(\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}_{ig}^{-\mathrm{T}}\Big] \widehat{\boldsymbol{\nabla}}\phi_{ig}\, \omega_{ig}\,,\end{aligned} \tag{2.80}$$

where each term is defined in the following methods into *elemOperStructure* class:

1. $\sum_{ig} \Big[\alpha \mathrm{J}^{-2/3}\Big(\mathbf{F} - \frac{1}{3}\mathrm{I}_1(\mathbf{C})\mathbf{F}^{-\mathrm{T}}\Big) e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\Big] \widehat{\boldsymbol{\nabla}}\phi_{ig}\, \omega_{ig}$: method source_P1iso_Exp;
2. $\sum_{ig} \mathrm{J}\frac{\kappa}{2}\Big[\mathrm{J} - 1 + \frac{1}{\mathrm{J}}\ln \mathrm{J}\Big)\mathbf{F}^{-\mathrm{T}}\Big] \widehat{\boldsymbol{\nabla}}\phi_{ig}\, \omega_{ig}$: method source_Pvol;

As done for the Neo-Hookean model, we obtain the linearization of the stiffness term with respect to $\mathbf{F}$, separating the isochoric and volumetric part:

$$\begin{aligned}\mathbb{C}_{\mathrm{iso}} : \delta\mathbf{F} \;\; &= -\frac{2}{3}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-2/3}(1 + \gamma \mathrm{I}_1(\overline{\mathbf{C}}))(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F} \\ &\quad + \frac{2}{9}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{I}_1(\overline{\mathbf{C}})(1 + \gamma \mathrm{I}_1(\overline{\mathbf{C}}))(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}} \\ &\quad - \frac{2}{3}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-2/3}(1 + \gamma \mathrm{I}_1(\overline{\mathbf{C}}))(\mathbf{F} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}} \\ &\quad + 2\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-4/3}(\mathbf{F} : \delta\mathbf{F})\mathbf{F} \\ &\quad + \alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-2/3}\delta\mathbf{F} \\ &\quad + \frac{2}{3}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{I}_1(\overline{\mathbf{C}})\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}\,.\end{aligned} \tag{2.81}$$

$$\begin{aligned}\mathbb{C}_{\mathrm{vol}} : \delta\mathbf{F} \;\; &= \frac{\kappa}{2}\mathrm{J}\Big(2\mathrm{J} - 1 + \frac{1}{J}\Big)(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}} \\ &\quad - \frac{\kappa}{2}(\mathrm{J}^2 - \mathrm{J} + \ln\mathrm{J})\mathbf{F}^{-\mathrm{T}}\delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}\,.\end{aligned} \tag{2.82}$$

where each term is defined in the following methods into *elemOperStructure* class:

1. $-\frac{2}{3}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-2/3}(1 + \gamma \mathrm{I}_1(\overline{\mathbf{C}}))(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}$: method stiff_Jac_P1iso_Exp_1term;
2. $\frac{2}{9}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{I}_1(\overline{\mathbf{C}})(1 + \gamma \mathrm{I}_1(\overline{\mathbf{C}}))(\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_P1iso_Exp_2term;
3. $-\frac{2}{3}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-2/3}(1 + \gamma \mathrm{I}_1(\overline{\mathbf{C}}))(\mathbf{F} : \delta\mathbf{F})\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_P1iso_Exp_3term;
4. $2\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-4/3}(\mathbf{F} : \delta\mathbf{F})\mathbf{F}$: method stiff_Jac_P1iso_Exp_4term;
5. $\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{J}^{-2/3}\delta\mathbf{F}$: method stiff_Jac_P1iso_Exp_5term;
6. $\frac{2}{3}\alpha e^{\gamma(\mathrm{I}_1(\overline{\mathbf{C}})-3)}\mathrm{I}_1(\overline{\mathbf{C}})\mathbf{F}^{-\mathrm{T}} : \delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_P1iso_Exp_6term;

7. $\frac{\kappa}{2}\mathrm{J}\Big(2\mathrm{J}-1+\frac{1}{J}\Big)(\mathbf{F}^{-\mathrm{T}}:\delta\mathbf{F})\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_Pvol_1term;

8. $-\frac{\kappa}{2}(\mathrm{J}^2-\mathrm{J}+\ln\mathrm{J})\mathbf{F}^{-\mathrm{T}}\delta\mathbf{F}^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}$: method stiff_Jac_Pvol_2term;

For further information [35].

2.3 The design of the structural solver

The architecture of the solver that has been implemented in the finite element library *LifeV*, which exploits parallelism, is shown in 3.3.

The core of the solver is the class *NonLinearStructureSolver* (NLSS) where we define the materials implemented and the principal methods to define the problem. NLSS uses *VenantKirchhoffSolver* (VKS) to define some variables and methods shared between linear and non-linear elasticity. Moreover into NLSS the template *nonLinRichardson* (NLR) is recalled

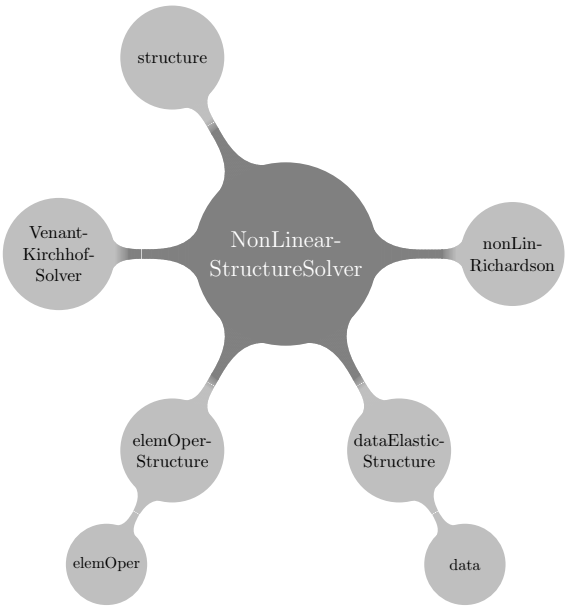


Figure 2.1: Structural solver: general view

to solve the nonlinear system (2.62). Into the classes *elemOper* (EO), *elemOperStructure* (EOS) the FE implementation of the materials are defined. Finally, the class *dataElastic-Structure* (DES) is the communicator from *data* file and NLSS, VKS and the file *structure*. *structure* is the main, where the methods defined in NLSS are recalled to solve the problem of non-linear elasticity. Into *structure* it is also possible to define the boundary conditions for a specific problem.

2.3.1 NLSS principal methods

NLSS is composed of some methods that are recalled from *structure* that is the main of the solver for nonlinear elasticity. The principal methods are setup, that initializes the parameters, buildSystem that computes the mass matrix $\mathbf{M}$ and the linear part of the stiffness term $\mathbf{K}_{\text{lin}}$, updateSystem that builds the right-hand side without boundary conditions of the problem, rhs_{nbc}, and iterate that recalls NLR. Althought other methods defined into NLSS are recalled internally. The most important methods from this point of view are updateNonlinearTerms that computes the nonlinear part of the stiffness term $\mathbf{K}_{\text{nl}}$ and $\mathbf{k}(\boldsymbol{\eta})$, updateNonlinearMatrix and computeMatrix that compute the non-linear part of stiffness term and assembles the last one respectively. These two methods are recalled by evalResidual that compute the Newton residual res_{R}. To build the Jacobian of the linearized system the methods updateJacobian that computes the Jacoabian and solveJacobian that assembles the Jacobian are used.
Finally, to calculate the kinematics variables used to build the nonlinear part of the stiffness term and to build the Jacobian of the linearized system, the method computeKinematicsVariables is used.

The list below summarizes the principal methods of NLSS and their functions.

- setup: Inizializes parameters;
- buildSystem : Computes the mass matrix and the linear part of the stiffness term $\mathbf{K}_{\text{lin}}$;
- initialize: Imposes the initial conditions $\boldsymbol{\eta}_0$ and $\dot{\boldsymbol{\eta}}_0$;
- updateSystem: Computes the right hand side (rhs_{nbc}) of the system without boundary conditions:
  $\text{rhs}_{\text{nbc}} = \frac{2}{\delta \text{t}^2}\mathbf{M}(\boldsymbol{\eta}^n + \delta \text{t}\dot{\boldsymbol{\eta}}^n) - (1-\zeta)\mathbf{K}_{\text{lin}}\boldsymbol{\eta}^n - (1-\zeta)\mathbf{K}_{\text{nl}}\boldsymbol{\eta}^n - (1-\zeta)\mathbf{k}(\boldsymbol{\eta}^n)$;
- updateNonlinearTerms: Computes the non-linear stiffness terms $\mathbf{K}_{\text{nl}}$, $\mathbf{k}(\boldsymbol{\eta}^n)$
- iterate: Recalls *nonLinRichardson* and go to the next time-step when the Newton convergence test is satisfied;
- evalResidual: Evaluates the Newton residual (res_{R}) defined as:
  $\text{res}_{\text{R}} = \frac{2}{\delta \text{t}^2}\mathbf{M}\boldsymbol{\eta}_k^{n+1} + \zeta(\mathbf{K}_{\text{lin}} + \mathbf{K}_{\text{nl}})\boldsymbol{\eta}_k^{n+1} + \zeta\mathbf{k}(\boldsymbol{\eta}_k^{n+1}) - \text{rhs}_{\text{nbc}} + \text{b.c.}$;
- computeMatrix: Assembles the stiffness terms $\zeta(\mathbf{K}_{\text{lin}}, \mathbf{K}_{\text{nl}})\boldsymbol{\eta}_k^{n+1}$, $\zeta\mathbf{k}(\boldsymbol{\eta}_k^{n+1})$;
- updateNonlinearMatrix: Computes the non-linear stiffness terms $\mathbf{K}_{\text{nl}}$, $\zeta\mathbf{k}(\boldsymbol{\eta}_k^{n+1})$;
- computeKinematicsVariables: Computes the kinematics variables $\mathbf{F}$, $\text{Cof}(\mathbf{F})$, $\text{Tr}(\overline{\mathbf{C}})$, $\text{Tr}(\mathbf{C})$, $\det(\mathbf{F})$;
- solveJacobian: Assembles the Jacobian of the linearized system calling updateJacobian, applies the boundary conditions calling applyBoundaryConditions and solve the linearized system calling P-GMRES;

2.3.2 Structural solver architecture

The figure 2.2 gives a general scheme that explains how the structural solver works and shows the interaction among the various components.

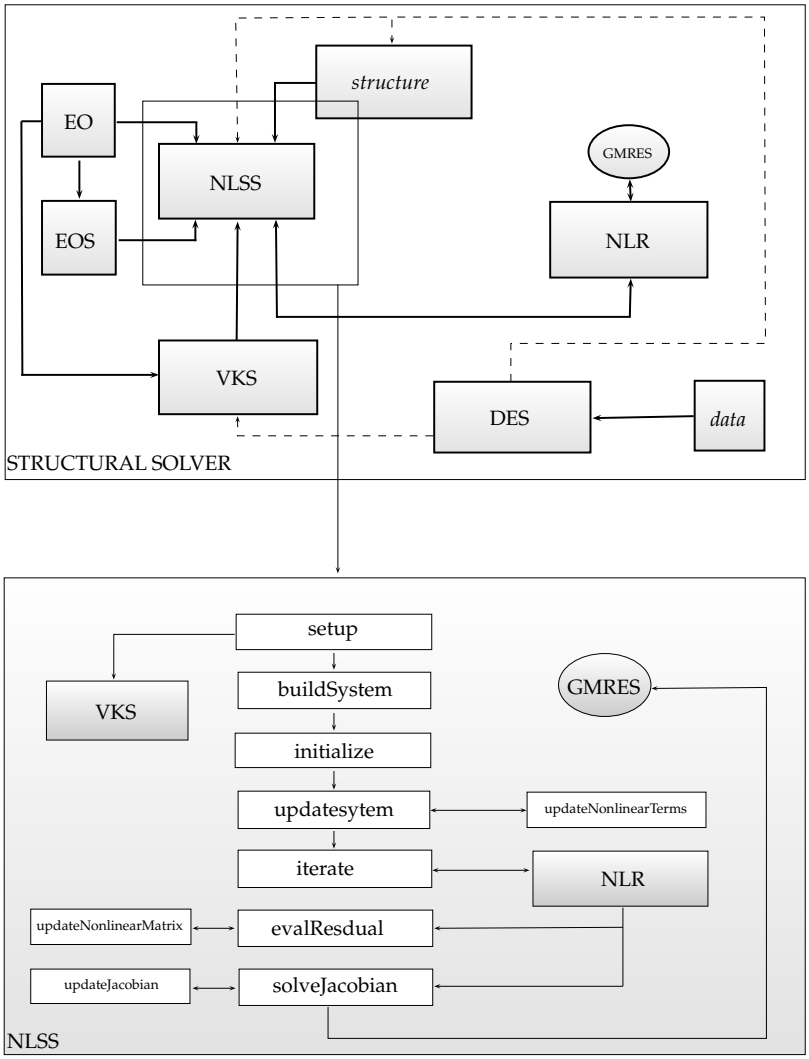


Figure 2.2: Structural solver design

2.4 The validation of the structural solver

In this section we perform a validation of the structural solver by comparing the results on academic test cases of which there is an analytical solution. We remarks that all materials implemented are isotropic.

2.4.1 Analytical test case

We consider the case of homogeneus pure strain of a cube. For this simple case exists an analitycal solution for some compressible hyperelastic materials. The problem reads:

$$\begin{aligned} &\widehat{\nabla} \cdot \mathbf{P} + \widehat{\rho}\,\mathbf{b} = \mathbf{0}\,, \qquad \text{in } [0,\,\mathrm{L}]^3\,, \\ &\boldsymbol{\eta}(\mathrm{x}=0) \cdot \mathbf{e}_{\mathrm{x}} = 0\,, \\ &\boldsymbol{\eta}(\mathrm{x}=\mathrm{L}) \cdot \mathbf{e}_{\mathrm{x}} = \overline{p}\,, \end{aligned} \tag{2.83}$$

where $\mathbf{e}_{\mathrm{x}}$ is the unit vector in the x direction and $\overline{p}$ is a prescribed pressure. The test is the normal traction of a cube in direction of the Cartesian axis x, with homogeneous Dirichlet condition in the same direction and on the opposite face. To correctly reproduce the elastostatic response (2.83) we have waited the end of the transitory of the elastodynamic system.

This simple case allows to find an analitycal expression for the first Piola-Kirchhoff stress tensor. In fact, it is possible to define the deformation tensor and the right Cauchy-Green tensor in terms of principal strain:

$$\mathbf{F} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \lambda_1^2 & 0 & 0 \\ 0 & \lambda_2^2 & 0 \\ 0 & 0 & \lambda_3^2 \end{bmatrix}. \tag{2.84}$$

In this simple case the strain-energy function only depends on deformation tensor $\mathbf{F}$. Hence we are allowed to write:

$$\mathrm{W} = \mathrm{W}(\lambda_1, \lambda_2, \lambda_3)\,. \tag{2.85}$$

Furthermore, for isotropic materials the strain-energy function is subject to the following simmetries:

$$\mathrm{W}(\lambda_1, \lambda_2, \lambda_3) = \mathrm{W}(\lambda_1, \lambda_3, \lambda_2) = \mathrm{W}(\lambda_3, \lambda_1, \lambda_2)\,. \tag{2.86}$$

The Jacobian of $\mathbf{F}$ is $\mathrm{J} = \lambda_1\,\lambda_2\,\lambda_3$ and because of the symmetry of the test case, $\lambda_2 = \lambda_3$. So we can write:

$$\lambda_2 = \lambda_3 = \sqrt{\left(\frac{\mathrm{J}}{\lambda_1}\right)}\,. \tag{2.87}$$

Hence, formally it is possible to write the following relation for the strain-energy function:

$$\mathrm{W} = \mathrm{W}\left(\lambda_1, \sqrt{\left(\frac{\mathrm{J}}{\lambda_1}\right)}\right). \tag{2.88}$$

From the last equation it is possible to calculate the first Piola-Kirchhoff stress tensor in an analytical way. Indeed, through the expression of P_{22} it is possible to compute the Jacobian J as a function of λ_1 only. Hence, the principal stress P_{11} is only a function of λ_1 and it is possible to calculate P_{11} from the numerical data and to compare it with the applied pressure on Neumann surface. The principal stress components we are interested (i.e. $\mathrm{P}_{11}, \mathrm{P}_{22}$) to become, for each material, are the following:

- *St. Venant-Kirchhoff*:

$$\begin{aligned} \mathrm{P}_{11} &= 2\mathrm{F}_{11}\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{C}_{11}}\Big) = \frac{\lambda}{2}\Big(\mathrm{I}_1(\mathbf{C}) - 3\Big)\lambda_1 - \mu\lambda_1 + \mu\lambda_1^3\,; \\ \mathrm{P}_{22} &= 2\mathrm{F}_{22}\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{C}_{22}}\Big) = \frac{\lambda}{2}\Big(\mathrm{I}_1(\mathbf{C}) - 3\Big)\lambda_1 - \mu\lambda_2 + \mu\lambda_2^3\,; \end{aligned} \tag{2.89}$$

- *Neo-Hookean*:

$$\begin{aligned} \mathrm{P}_{11} &= 2\mathrm{F}_{11}\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{C}_{11}}\Big) = \mu \mathrm{J}^{-2/3}\Big(\lambda_1 - \frac{\mathrm{I}_1(\mathbf{C})}{3\lambda_1}\Big) + \frac{\kappa}{2}\big(\mathrm{J}^2 - \mathrm{J} + \log(\mathrm{J})\big)\frac{1}{\lambda_1}\,; \\ \mathrm{P}_{22} &= 2\mathrm{F}_{22}\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{C}_{22}}\Big) = \mu \mathrm{J}^{-2/3}\Big(\lambda_2 - \frac{\mathrm{I}_1(\mathbf{C})}{3\lambda_2}\Big) + \frac{\kappa}{2}\big(\mathrm{J}^2 - \mathrm{J} + \log(\mathrm{J})\big)\frac{1}{\lambda_2}\,; \end{aligned} \tag{2.90}$$

- *Exponential*:

$$\begin{aligned} \mathrm{P}_{11} = 2\mathrm{F}_{11}\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{C}_{11}}\Big) &= \alpha \exp\big[\mathrm{I}_1(\overline{\mathbf{C}}) - 3\big]\mathrm{J}^{-2/3}\Big(\lambda_1 - \frac{\mathrm{I}_1(\mathbf{C})}{3\lambda_1}\Big) + \\ &\quad + \frac{\kappa}{2}\big(\mathrm{J}^2 - \mathrm{J} + \log(\mathrm{J})\big)\frac{1}{\lambda_1}\,; \\ \mathrm{P}_{22} = 2\mathrm{F}_{22}\Big(\frac{\partial \mathrm{W}}{\partial \mathrm{C}_{22}}\Big) &= \alpha \exp\big[\mathrm{I}_1(\overline{\mathbf{C}}) - 3\big]\mathrm{J}^{-2/3}\Big(\lambda_2 - \frac{\mathrm{I}_1(\mathbf{C})}{3\lambda_2}\Big) + \\ &\quad + \frac{\kappa}{2}\big(\mathrm{J}^2 - \mathrm{J} + \log(\mathrm{J})\big)\frac{1}{\lambda_2}\,; \end{aligned} \tag{2.91}$$

The second relation of (2.89), (2.90), (2.91) is equal to zero. Therefore, we can calculate from it the Jacobian J and from the first expression of (2.89), (2.90), (2.91) we can compute P_{11} which now is only dependent on λ_1. Then it is possible to compare the applied pressure (Neumann condition) with P_{11} calculated from the numerical results in terms of λ_1.

2.4.2 Numerical test case

The test case for the validation of the code uses a structured mesh composed of tetrahedra with 125 nodes. The finite elements used are the first order polynomial (P1) hence the nodes of the mesh are also the degrees of freedom. To perform the validation of structural solver we have taken the numerical results in terms of principal strain λ_1, λ_2, λ_3. Then we have calculated the component P_{11} of the first Piola-Kirchhoff stress tensor from the analytical expressions reported in the previous paragraph. Finally, the Neumann condition applied to the cube has compared with P_{11} calculated from the numerical results. Further we have computed the relative error as:

$$\mathrm{err} = \frac{\mathrm{P}_{11_{\mathrm{an}}} - \mathrm{P}_{11_{\mathrm{num}}}}{\mathrm{P}_{11_{\mathrm{an}}}}\,. \tag{2.92}$$

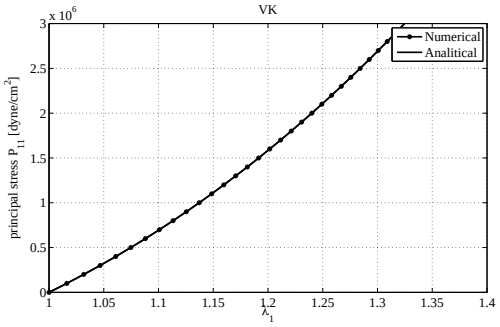
Table 2.2: Set of structural parameters used for validation

	$\mathbf{Set}_1$	$\mathbf{Set}_2$	$\mathbf{Set}_3$	u.d.m
κ	1e8	5e8	1e9	[dyne/cm^2]
ν	0.45	0.45	0.45	-
E	6e6	7e6	8e6	[dyne/cm^2]
α	2e6	2.5e6	3e6	[dyne/cm^2]
γ	0.80	0.75	0.70	-

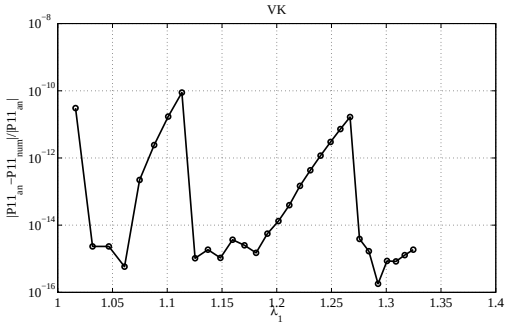
We have choosen three sets of structural parameters (table 2.2) to evaluate the sensitivity of the solver to the variation of structural data for each structural model. The three set used are typical values for biological tissues [9]-[25]-[12]-[11].

In the figures 2.3, 2.4, 2.5 for the first set of structural parameters we observe that the analitycal and the numerical solution are overlapped. In particular the relative error is less than 10^{10}.

Using the second and third sets of structural parameters we obtain the same results in terms of error. In particular, also in this case the relative error is less than the Newton tolerance and the numerical curve is overlapped to the analytical solution. Hence the accurancy of the solution is not sensitive to the choice of structural parameters, with a reasonable range. Finally in figure 2.12 we show the difference between three structural models. We recognize the correct trend for the 3 models considered [25].

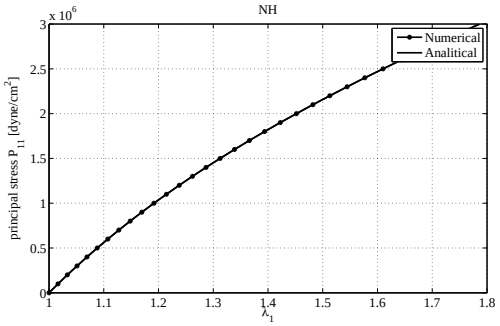


(a) P_{11} vs. λ_1: Numerical vs. Analytical

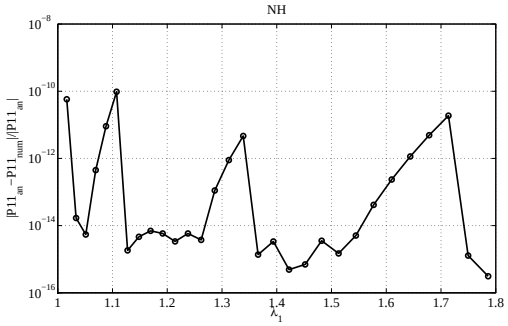


(b) Relative error

Figure 2.3: St.Venant-Kirchhoff, first set of structural parameters

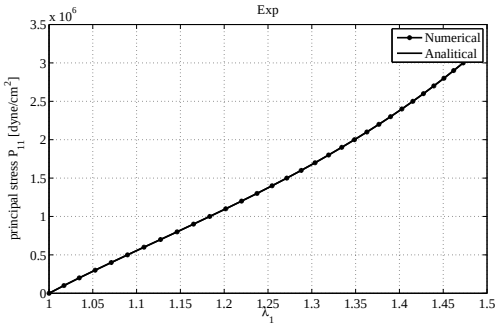


(a) P_{11} vs. λ_1: Numerical vs. Analytical

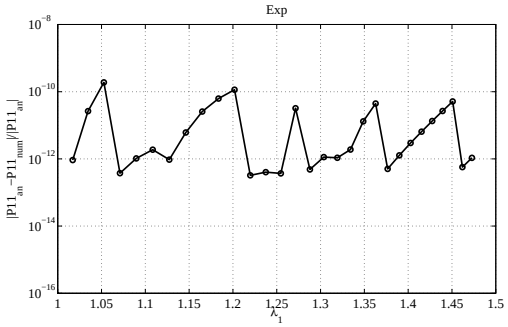


(b) Relative error

Figure 2.4: Neo-Hookean, first set of structural parameters

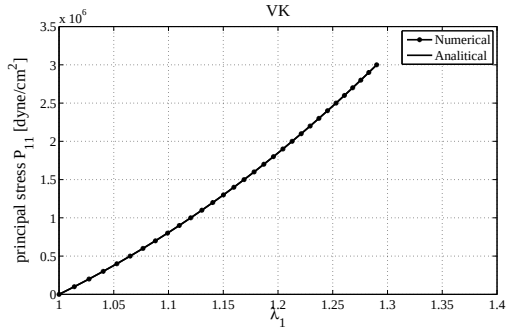


(a) P_{11} vs. λ_1: Numerical vs. Analytical

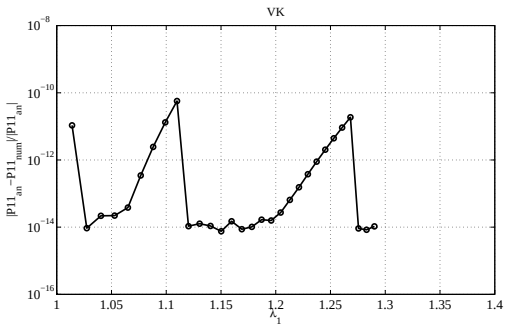


(b) Relative error

Figure 2.5: Exponential, first set of structural parameters

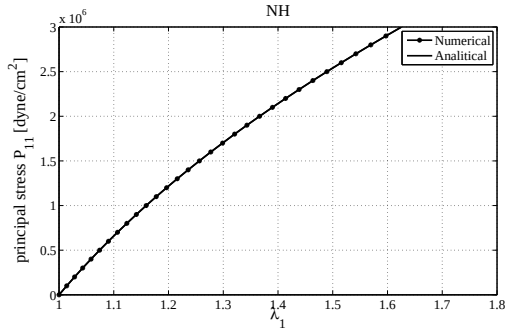


(a) P_{11} vs. λ_1: Numerical vs. Analytical

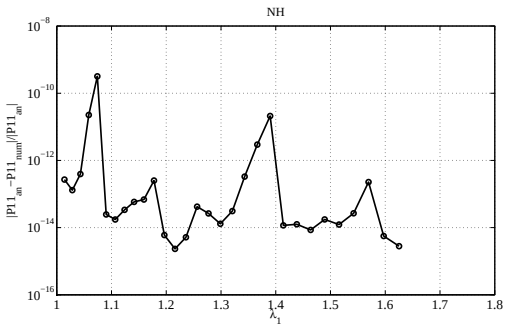


(b) Relative error

Figure 2.6: St.Venant-Kirchhoff, second set of parameters

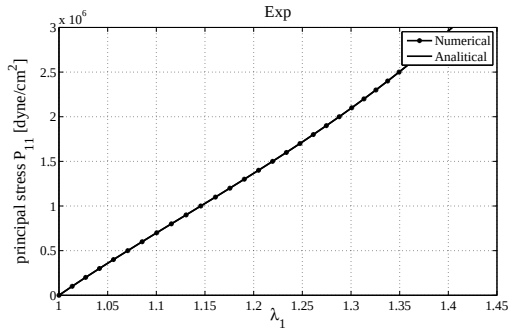


(a) P_{11} vs. λ_1: Numerical vs. Analytical

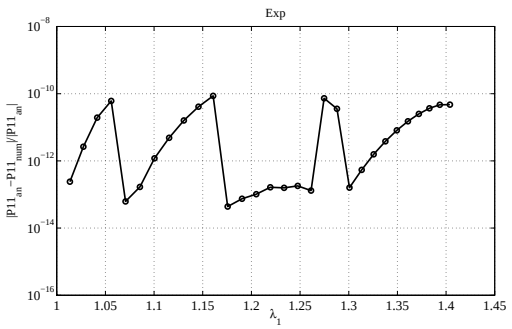


(b) Relative error

Figure 2.7: Neo-Hookean, second set of parameters

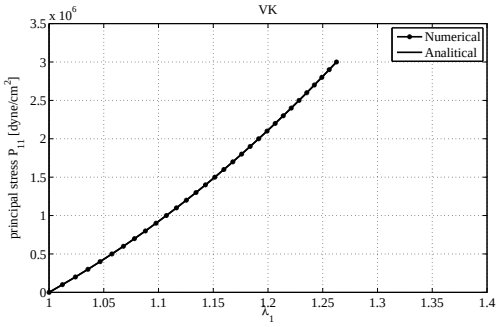


(a) P_{11} vs. λ_1: Numerical vs. Analytical

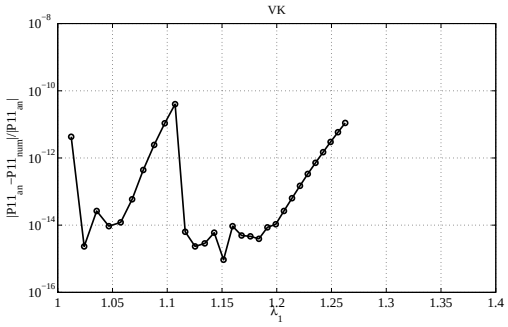


(b) Relative error

Figure 2.8: Exponential, second set of parameters

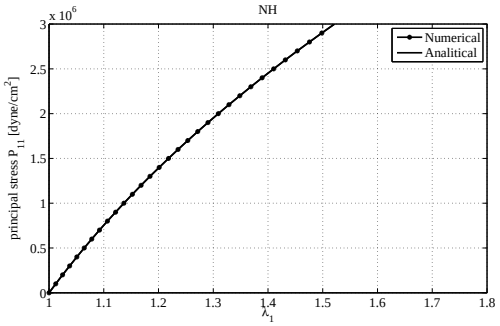


(a) P_{11} vs. λ_1: Numerical vs. Analytical

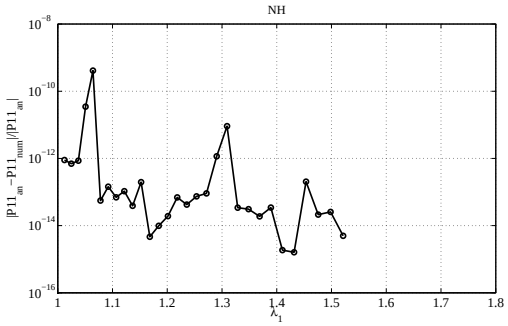


(b) Relative error

Figure 2.9: St.Venant-Kirchhoff, third set of parameters

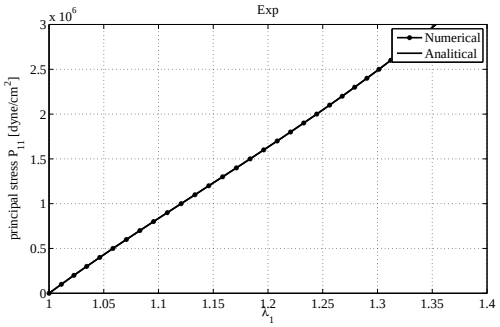


(a) P_{11} vs. λ_1: Numerical vs. Analytical

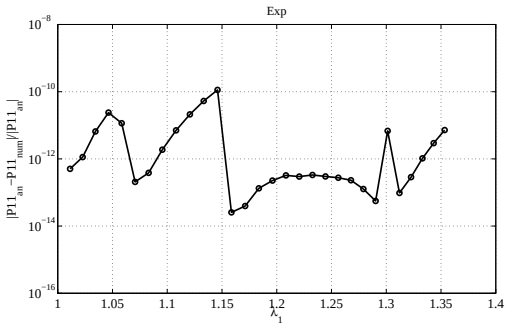


(b) Relative error

Figure 2.10: Neo-Hookean, third set of parameters

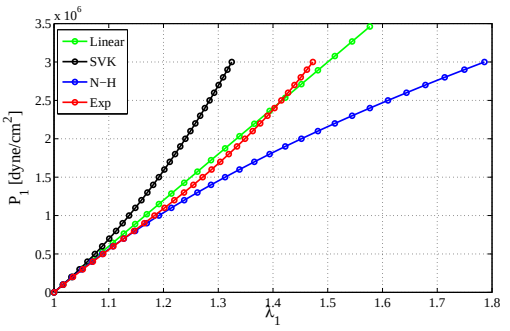


(a) P_{11} vs. λ_1: Numerical vs. Analytical

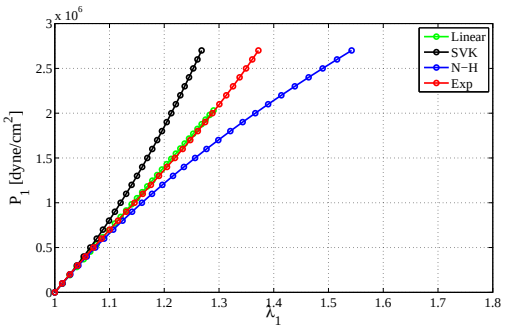


(b) Relative error

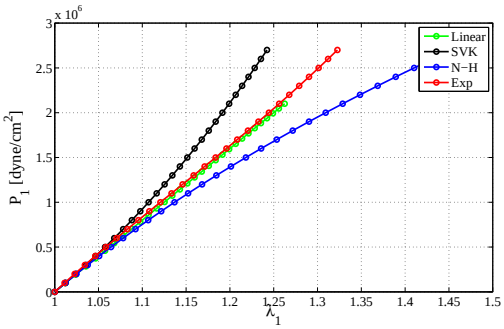
Figure 2.11: Exponential, third set of parameters



(a) Comparison set$_1$



(b) Comparison set$_2$



(c) Comparison set$_3$

Figure 2.12: Comparison between three nonlinear structural models and linear elasticity: P_{11} vs. λ_1

2.5 Hollow cylinder inflation test

We have also created an inflation test of a hollow cylinder. In particular we have fixed the structural parameters equal to set$_1$ (table 2.3) in accord with the common data used for structural simulations of biological tissues [25]-[31].

Table 2.3: Set of structural parameters used for hollow cylinder inflation test

ρ_s[g/cm^3]	κ[dyne/cm^2]	ν[ ]	E[dyne/cm^2]	α[dyne/cm^2]	γ[ ]
1.2	1×10^8	0.45	6×10^6	2×10^6	0.8

The test involves the application of an internal hydrostatic pressure as a function of time only, with the following expression:

$$\mathrm{p_{in}} = 260000\sin(40\pi \mathrm{t}) \tag{2.93}$$

where t is the time and $\mathrm{p_{in}}$ is measured in [dyne/cm^2]. In addition to the internal pressure, Dirichlet boundary conditions on the basis, parallel to the axis of the hollow cylinder and

Neumann boundary conditions on the external surface in the radial direction are applied. Formally, we have solved the following problem:

$$\begin{cases} \widehat{\boldsymbol{\nabla}} \cdot \mathbf{T}_{\mathrm{s}} = 0\,, \\ \mathbf{T}_{\mathrm{s}}(\mathrm{R_{in}}) = \mathrm{P_{in}}\mathbf{e}_{\mathrm{R}}\,, \\ \mathbf{T}_{\mathrm{s}}(\mathrm{R_{out}}) = 0\,, \\ \boldsymbol{\eta}(\mathrm{z} = -\mathrm{L}/2) \cdot \mathbf{e}_{\mathrm{Z}} = 0\,, \\ \boldsymbol{\eta}(\mathrm{z} = \mathrm{L}/2) \cdot \mathbf{e}_{\mathrm{Z}} = 0\,, \end{cases} \tag{2.94}$$

where, $\mathbf{e}_{\mathrm{R}}$ and $\mathbf{e}_{\mathrm{Z}}$ are the unit vectors associated to the radial and axial directions.
The mesh used is *vessel22* and the finite-element discretization is performed by P1 elements. In table 2.4 it is possible to see the mesh properties in details.

Table 2.4: Mesh properties of *vessel22*

Nodes	**Triangles**	**Thetrahedra**	**Length**	**Inner radius**	**Outer radius**
3220	2912	14784	5 cm	0.5 cm	0.6 cm

Moreover the time interval of the test is between 0 and 0.025 seconds with a time-step of 0.001 seconds. Hence the temporal steps of the simulation are 25. For this academic test it is possible to derive an analytical solution for linear elasticity only. Then, we can compare, from a qualitative point of view, the solution obtained by nonlinear material models to evalute if the results are reasonable in modulus and shape. The graphs obtained in the last part of the results (section 2.5.2) are obtained by meaning the displacement of the internal nodes of the mid section of the hollow cylinder as showed in figure 2.5

(a) Isometric visual of the mid section (b) Plane visual of the mid section

Figure 2.13: Visual of the mid section of the hollow cylinder for the structural analysis

2.5.1 Linear elasticity: analytical solution

The test case can be effectively described in a cylindrical coordinate system, given the geometry of the problem shown in figure 2.5.1.

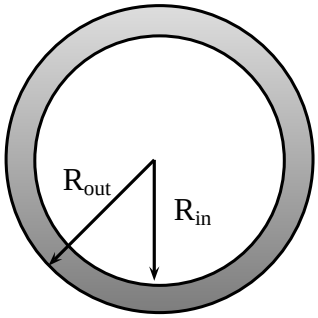


Figure 2.14: Normal section of the hollow cylinder

In particular, for linear elasticity, the deformation gradient $\mathbf{F}$ is equal to identity. Furthermore, no distinction is made between the current configuration and reference configuration as these are considered infinitesimal deformations. Therefore, the system of cylindrical coordinates adopted is just one, $(\widehat{\mathrm{R}}, \widehat{\Theta}, \widehat{\mathrm{Z}})$. Since the gradient of deformation equal the identity matrix, it follows that the Green-Lagrange tensor is:

$$\mathbf{E} = \frac{1}{2}\Big(\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta} + \widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}\Big)\,, \tag{2.95}$$

while, the constitutive law, valids for isotorpic materials, is:

$$\mathbf{T}_{\mathrm{s}} = \lambda \mathrm{Tr}(\mathbf{E}) + 2\mu\mathbf{E}\,, \tag{2.96}$$

where μ and λ are the Lamè constants introduced in (2.44)-(2.43). We have to solve the problem (2.94), here reported:

$$\begin{cases} \widehat{\boldsymbol{\nabla}} \cdot \mathbf{T}_{\mathrm{s}} = 0\,, \\ \mathbf{T}_{\mathrm{s}}(\mathrm{R_{in}}) = \mathrm{P_{in}}\mathbf{e}_{\mathrm{R}}\,, \\ \mathbf{T}_{\mathrm{s}}(\mathrm{R_{out}}) = 0\,, \\ \boldsymbol{\eta}(\mathrm{z} = -\mathrm{L}/2) \cdot \mathbf{e}_{\mathrm{Z}} = 0\,, \\ \boldsymbol{\eta}(\mathrm{z} = \mathrm{L}/2) \cdot \mathbf{e}_{\mathrm{Z}} = 0\,. \end{cases}$$

To determine the analytical solution, we assume that the displacement field $\boldsymbol{\eta} = (\eta_{\widehat{\mathrm{R}}}, \eta_{\widehat{\Theta}}, \eta_{\widehat{\mathrm{Z}}})$, has only one non-zero component, namely the component $\eta_{\widehat{\mathrm{R}}}$, and that it depends only on the radial coordinate $\widehat{\mathrm{R}}$. Then, measuring the gradient of the displacement field $\widehat{\boldsymbol{\nabla}}\boldsymbol{\eta}$ in cylindrical coordinates, it is possible to calculate the Green-Lagrange tensor $\mathbf{E}$. The non-zero

components of the Cauchy stress tensor are then:

$$\begin{cases} \mathbf{T}_{\mathrm{s},\widehat{R}\widehat{R}} = (2\mu+\lambda)\mathrm{U}'(\widehat{R}) + \dfrac{\lambda}{\widehat{R}}\mathrm{U}(\widehat{R}) \\ \mathbf{T}_{\mathrm{s},\widehat{\Theta}\widehat{\Theta}} = \dfrac{1}{\widehat{R}}(2\mu+\lambda)\mathrm{U}(\widehat{R}) + \lambda\mathrm{U}'(\widehat{R}) \\ \mathbf{T}_{\mathrm{s},\widehat{Z}\widehat{Z}} = \lambda\Big(\mathrm{U}'(\widehat{R}) + \dfrac{\mathrm{U}(\widehat{R})}{\widehat{R}}\Big) \end{cases} \tag{2.97}$$

By writing the balance equation of momentum in cylindrical coordinates and substituting the components of the Cauchy tensor just obtained (2.97), we obtain the resolvent equation:

$$\frac{\partial \mathbf{T}_{\mathrm{s},\widehat{\mathrm{R}}\widehat{\mathrm{R}}}}{\partial \widehat{\mathrm{R}}} + \frac{1}{\widehat{\mathrm{R}}}(\mathbf{T}_{\mathrm{s},\widehat{\mathrm{R}}\widehat{\mathrm{R}}} - \mathbf{T}_{\mathrm{s},\widehat{\Theta}\widehat{\Theta}}) = 0\,, \tag{2.98}$$

that leads to an equi-dimensional or Euler-Cauchy equation:

$$\widehat{\mathrm{R}}^2\mathrm{U}''(\widehat{\mathrm{R}}) + \widehat{\mathrm{R}}\mathrm{U}'(\widehat{\mathrm{R}}) - \mathrm{U}(\widehat{\mathrm{R}}) = 0 \tag{2.99}$$

with solution:

$$\mathrm{U}(\widehat{\mathrm{R}}) = \mathrm{C}_1\widehat{\mathrm{R}} + \frac{\mathrm{C}_2}{\widehat{\mathrm{R}}} \tag{2.100}$$

The constants of integration are determined by imposing the boundary conditions defined in (2.94) and have the following form:

$$\begin{cases} \mathrm{C}_1 = \dfrac{1}{2(\mu+\lambda)}\left(\dfrac{\widehat{\mathrm{R}}_{\mathrm{in}}^2\mathrm{P}_{\mathrm{in}} - \widehat{\mathrm{R}}_{\mathrm{out}}^2\mathrm{P}_{\mathrm{out}}}{\widehat{\mathrm{R}}_{\mathrm{out}}^2 - \widehat{\mathrm{R}}_{\mathrm{in}}^2}\right) \\ \mathrm{C}_2 = \dfrac{\widehat{\mathrm{R}}_{\mathrm{in}}^2\widehat{\mathrm{R}}_{\mathrm{out}}^2}{2\mu}\left(\dfrac{\mathrm{P}_{\mathrm{in}} - \mathrm{P}_{\mathrm{out}}}{\widehat{\mathrm{R}}_{\mathrm{out}}^2 - \widehat{\mathrm{R}}_{\mathrm{in}}^2}\right) \end{cases} \tag{2.101}$$

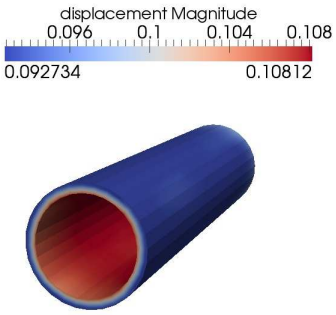
The displacement field then becomes:

$$\begin{cases} \eta_{\widehat{\mathrm{R}}} = \dfrac{\widehat{\mathrm{R}}}{2(\mu+\lambda)}\left(\dfrac{\widehat{\mathrm{R}}_{\mathrm{in}}^2\mathrm{P}_{\mathrm{in}} - \widehat{\mathrm{R}}_{\mathrm{out}}^2\mathrm{P}_{\mathrm{out}}}{\widehat{\mathrm{R}}_{\mathrm{out}}^2 - \widehat{\mathrm{R}}_{\mathrm{in}}^2}\right) + \dfrac{\widehat{\mathrm{R}}_{\mathrm{in}}^2\widehat{\mathrm{R}}_{\mathrm{out}}^2}{2\mu\widehat{\mathrm{R}}}\left(\dfrac{\mathrm{P}_{\mathrm{in}} - \mathrm{P}_{\mathrm{out}}}{\widehat{\mathrm{R}}_{\mathrm{out}}^2 - \widehat{\mathrm{R}}_{\mathrm{in}}^2}\right) \\ \eta_{\widehat{\Theta}} = 0 \\ \eta_{\widehat{\mathrm{Z}}} = 0 \end{cases} \tag{2.102}$$

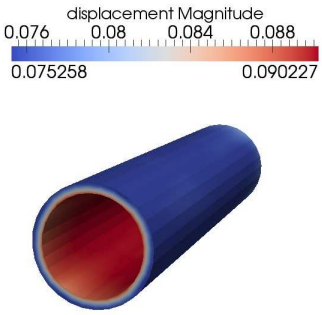
2.5.2 Results

In figure 2.15 we show the results at t = 0.012 s when the internal pressure is maximal. The colors represent the modulus of the displacement and each figure shows a different

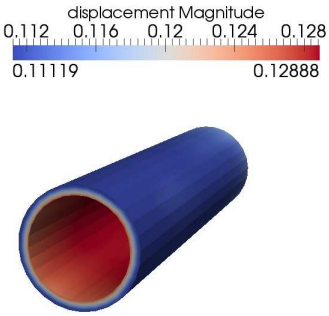
material. More precisely, figure 2.15(a) shows linear elasticity and it has an intermediate value of displacement magnitude[5], between the four materials, of $|\boldsymbol{\eta}_{\text{Lin}}| = 0.10812\,\text{cm}$, figure 2.15(b) is the St.Venant-Kirchhoff material and it has the smallest displacement of $|\boldsymbol{\eta}_{\text{SVK}}| = 0.090227\,\text{cm}$, figure 2.15(c) is the Neo-Hookean material and it has the largest displacement, $|\boldsymbol{\eta}_{\text{NH}}| = 0.12888\,\text{cm}$, finally, figure 2.15(d) is the Exponential material and it has an indermediate value of displacement, $|\boldsymbol{\eta}_{\text{Exp}}| = 0.117561\,\text{cm}$.



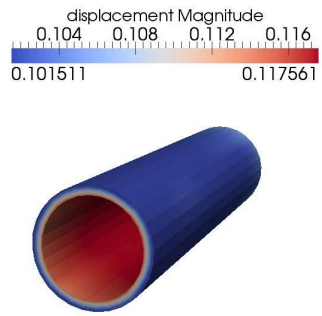
(a) Linear elasticity



(b) St.Venant-Kirchhoff



(c) Neo-Hookean



(d) Exponential

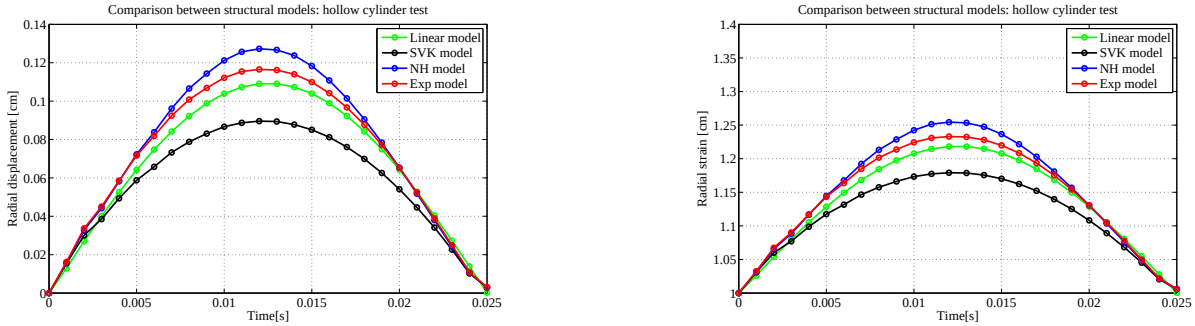
Figure 2.15: Comparison between three nonlinear structural models and linear elasticity, global view

Remark 5. Note how the materials response are consistent with the ones obtained on the test on the cube. In fact, we see that the nonlinear model of St.Venant-Kirchhoff is the most rigid, while Neo-Hookean model, shows the largest displacement with equal load applied. Finally, linear elasticity and nonlinear exponential model show an intermediate displacement.

[5]where the magnitude of the displacement is defined as $|\boldsymbol{\eta}| = \sqrt{\eta_{\text{x}}^2 + \eta_{\text{y}}^2 + \eta_{\text{z}}^2}$

Note that the trend of the solution, in terms of displacement, along the radius and the circumference is reasonable.

In figure 2.16 we show a comparison of radial displacement of an internal node of the hollow cylinder in its mid section, showed in the figure 2.5 versus time for four materials. In particular, the results reflect what has been seen in previous figures. In particular, the Neo-Hookean material presents the largest radial displacement while the St.Venant-Kirchhoff material has the smallest radial displacement.

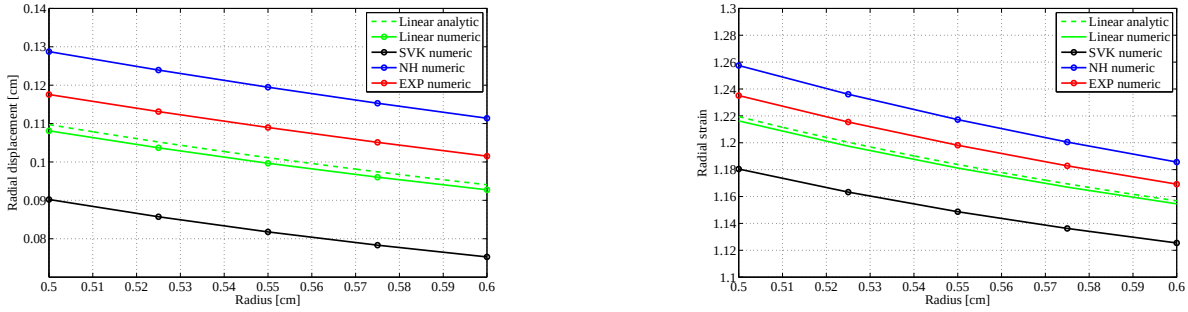


(a) Radial displacement

(b) Strain

Figure 2.16: Comparison of the radial displacement [cm] and strain vs. time [s] between 3 nonlinear structural models and linear elasticity for hollow cylinder inflation test

We have compared the numerical and analytical results for the linear material, superimposing the results of the three nonlinear materials. It is therefore possible to assess qualitatively if the values obtained in this test are reasonable. In figure 2.17 we show the radial displacement 2.17(a) and the radial strain 2.17(b) as a function of undeformed radius.



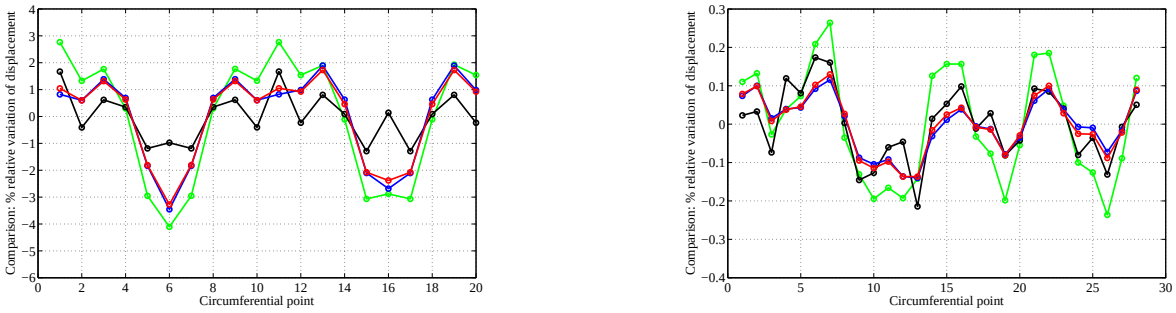
(a) Radial displacement

(b) Strain

Figure 2.17: Comparison of the radial displacement [cm] and strain vs. undeformed radius [cm] between 3 nonlinear structural models and linear elasticity for hollow cylinder inflation test

For the nonlinear materials here considered, an analytical solution for this test does not exist. Hence, it is not possible to have a quantitative validation of these results. However,

the results of the inflation test are qualitatively correct for both modulus and shape. In fact, if we see the simmetry of the displacement around the cirmumference of the hollow cylinder 2.18, we obtain errors less than 3% using a very coarse mesh (as shown in figure 2.18(a)) and error less than 0.2% using a fine mesh (as shown in figure 2.18(b)), for all nonlinear models. These errors are due to the not perfect symmetry of the meshes used.



(a) Mesh very coarse

(b) Mesh fine

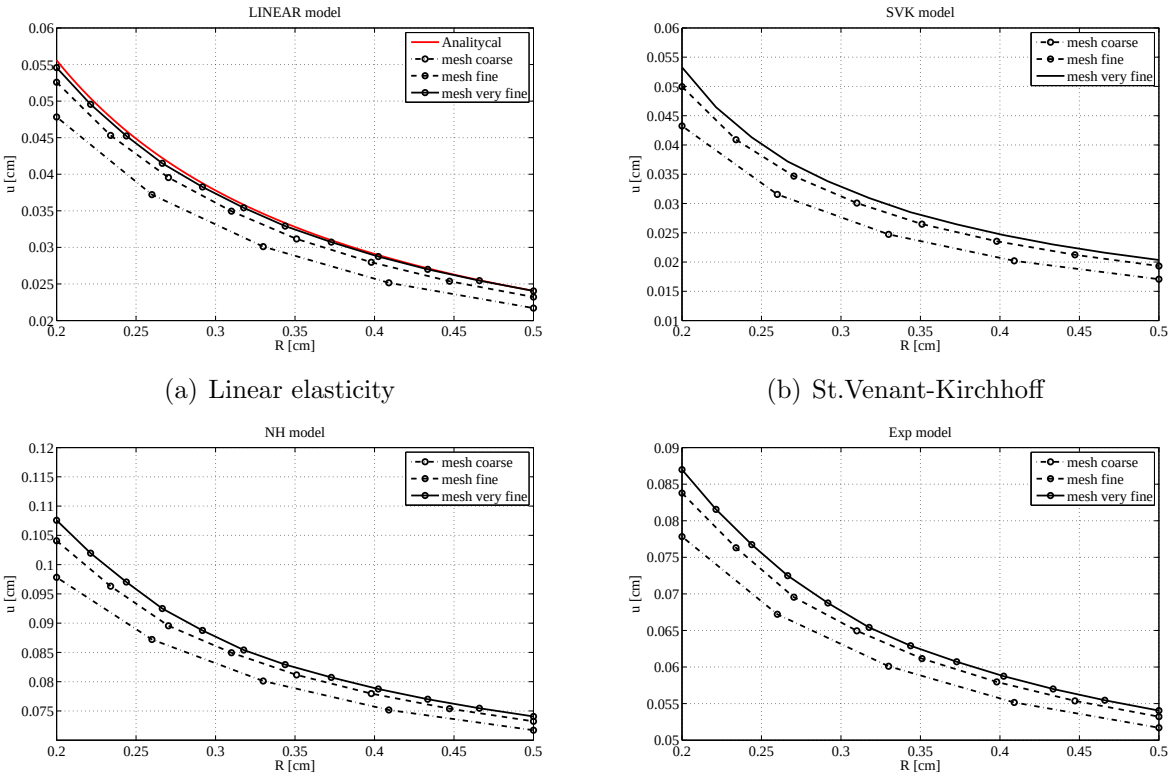
Figure 2.18: Simmetry percentual error for linear structural model and three nonlinear structural models using two different meshes

We can also see, in table 2.5, the mean value and the standard deviation of the displacement for each node around the circumference.

Table 2.5: Mean(cm) and standard deviation(cm) of the radial displacement for each node around the circumference using two different meshes

	LIN	**SVK**	**NH**	**Exp**
Mean [cm] (very coarse)	0.1061	0.0881	0.1262	0.1153
Standard deviation[cm] (very coarse)	0.0024	7.9550e-04	0.0021	0.0018
Mean [cm] (fine)	0.1080	0.0902	0.1287	0.1175
Standard deviation[cm] (fine)	1.5805e-04	8.4974e-05	9.3832e-05	9.0698e-05

Finally we show a mesh convergence test using three different meshes. It is possible to see in figure 2.19 that the radial displacement vs. undeformed radius (for all materials) has a stiffen behaviour using a mesh coarse.



(a) Linear elasticity

(b) St.Venant-Kirchhoff

(c) Neo-Hookean

(d) Exponential

Figure 2.19: Mesh convergence test for hollow cylinder inflation test

Chapter 3

FSI model

A correct construction of the FSI model, requires an accurate description of the fluid and of the structure. Blood can to be modeled as an incompressible Newtonian fluid while the wall can to be modeled as a relatively thin structure. As previously remarked, the most important part of this work was to implement nonlinear structural models to describe the arterial wall mechanics.

It is well-known [16] that an FSI problem is highly nonlinear, even when using linear elasticity. Indeed:

1. the position of the interface between fluid and structure is an unknown of the problem $\longrightarrow$ geometrical nonlinearity;

2. the convective term of the fluid problem is nonlinear. Moreover it depends also on the velocity of the fluid domain and thus on the structural displacements, because of the coupling conditions.

The integration of the nonlinear structural models into an FSI model adds a further nonlinearity to the problem. However it allows to describe the behaviour of the arterial wall and his interaction with blood flow (also in regime of large deformations) more precisely. In particular, it was shown that structural models such as Neo-Hookean and Exponential can be used to describe the arterial wall [36], [37] also in a multi-mechanism contest and using fibers[11].

Thus, it is interesting to use these models to describe the arterial wall mechanics in a FSI framework to better describe the typical haemodynamic indicators. The use of the St.Venant-Kirchhoff model, is instead justified, because it is very similar to the linear model and can be used as a benchmark for more complex models. We remark that this work is only a first step toward more detailed models that involves for example, fibers and pre-stress.

The purpose of this chapter is to introduce the equations of the FSI problem and to describe the numerical strategies to solve it. In particular we focus the attention on the integration of the NLSS into FSI solver. Finally, we present a comparison between linear elasticity and nonlinear elastic models on an academic FSI test case, with the objective of verifying the algorithm.

3.1 Fluid-structure interaction problem

Let us consider a mechanical system composed of a flow interacting with a deformable structure. In practice, it is necessary to calculate the solution in a time-varying computational domain. Its evolution is not known a priori but is obtained by solving an FSI problem. The domain Ω is divided into two subdomains Ω_s, occupied by the structure and Ω_f occupied by the fluid. Both are varying in time. The interface Σ between fluid and structure is defined as $\partial\Omega_s \cap \partial\Omega_f$ and is the boundary of the two subdomains.

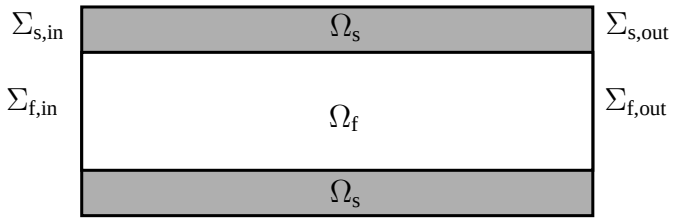


Figure 3.1: A longitudinal view of a typical domain of interest

For the typical geometries of interest the domain is shown in 3.1. On the boundary of Ω we can identify different layers. In particular $\partial\Omega \cap \partial\Omega_f = \Sigma_{f,in} \cup \Sigma_{f,out}$ and, where $\Sigma_{f,in}$ is the "inlet"or better the proximal fluid boundary, which correspond to the section nearer to the heart. Conversely, $\Sigma_{f,out}$ is the "outlet", or better distal fluid section.
We have put the words "outlet"and "inlet"into quotes, because even if the mean blood flow during a cardiac cycle in arteries is moving from the heart to perifery, we cannot exclude a priori the presence of reverse flow at certain instants.
The proximal and distal sections of the structural part are indicated by $\Sigma_{s,in}$ and $\Sigma_{s,out}$ respectively. The remaining portion of $\partial\Omega$, all on the structural side, are indicated by $\Sigma_{s,w}$. We wish to recall that the section Σ_{in} and Σ_{out} are artificial in the sense that they do not correspond to a physical separation but they are introduced to delimit the portion of artery under study. The issue of setting up appropriate conditions on these sections is rather complex and some discussion may be found in [38]. In this work, since we are focusing on the structural solver and on the FSI coupling we will use standard boundary conditions of Dirichlet and Neumann type. However, the technique here developed may be extended to treat more complex boundary conditions on those sections.

Furthermore we indicate with $\widehat{\Omega}$ a reference configuration. In particular, $\widehat{\Omega}_s$ is the reference configuration for the structure, which is assumed to be a natural configuration (i.e. a zero-stress configuration). This is questionable since it is well known that arteries are pre-stressed. However, in this work we have decided to neglect this aspect, which may be taken into account in a future extension. To describe the evolution of the domain we define two maps, one for the fluid domain and one for the solid domain. In particular the fluid map $\mathcal{A}$ is defined as:

$$\mathcal{A} : \widehat{\Omega}_\mathrm{f} \times (0, \mathrm{T}) \to \Omega_\mathrm{f}^\mathrm{t}\,, \qquad (\widehat{\mathbf{x}}, \mathrm{t}) \to \mathbf{x} = \mathcal{A}(\widehat{\mathbf{x}}, \mathrm{t})\,, \tag{3.1}$$

while the solid map, in accordance with Chapter 2, is defined as:

$$\mathcal{L} : \widehat{\Omega}_{\mathrm{s}} \times (0, \mathrm{T}) \to \Omega_{\mathrm{s}}^{\mathrm{t}}\,, \qquad (\widehat{\mathbf{x}}, \mathrm{t}) \to \mathbf{x} = \mathcal{L}(\widehat{\mathbf{x}}, \mathrm{t})\,. \tag{3.2}$$

The union of these two maps defines a homeomorphism on Ω, with the following interface condition:

$$\mathcal{L}^{\mathrm{t}} = \mathcal{A}^{\mathrm{t}} \qquad \text{on } \Sigma^{\mathrm{t}},\ \forall\, t \in (0, \mathrm{T})\,. \tag{3.3}$$

As previously stated, to describe the kinematics of the structure, we have used a Lagrangian approach. The map $\mathcal{L}^{\mathrm{t}}$ takes the following expression:

$$\mathcal{L}^{\mathrm{t}} = \widehat{\mathbf{x}} + \boldsymbol{\eta}(\widehat{\mathbf{x}}, \mathrm{t})\,, \qquad \widehat{\mathbf{x}} \in \widehat{\Omega}_s\,. \tag{3.4}$$

Instead, to describe the fluid map, we build a suitable extension of its value on the interface provided by the condition (3.3):

$$\mathcal{A}^{\mathrm{t}}(\widehat{\mathbf{x}}) = \widehat{\mathbf{x}} + \mathrm{Ext}(\boldsymbol{\eta}(\widehat{\mathbf{x}}, \mathrm{t})|_{\widehat{\Sigma}})\,, \qquad \widehat{\mathbf{x}} \in \widehat{\Omega}_f\,, \tag{3.5}$$

where, for instance, "Ext" is the harmonic extension operator in the reference configuration[39]. In particular, the choice of the extension operator is rather arbitrary as long as (3.3) is satisfied and the artificial boundaries are respected. For this reason $\mathcal{A}^{\mathrm{t}}$ is called Arbitrary Lagrangian Eulerian (ALE) map. For each function $\widehat{\mathrm{g}} : \widehat{\Omega}_s \to \mathbb{R}$ defined in the solid reference configuration, we denote by $\mathrm{g} = \widehat{\mathrm{g}} \circ (\mathcal{L})^{-1}$ the same function in the current configuration. In particular, we have the following relation between reference and current solid configuration:

$$\mathrm{g}(\mathbf{x}, \mathrm{t}) = \widehat{\mathrm{g}}((\mathcal{L}^{\mathrm{t}})^{-1}(\mathbf{x}), \mathrm{t})\,, \qquad \mathbf{x} \in \Omega_s\,. \tag{3.6}$$

Similarly for the fluid domain, given a function $f : \Omega_{\mathrm{f}}^{\mathrm{t}} \times (0, \mathrm{T}) \to \mathbb{R}$ defined in the current (Eulerian) configuration, the relation $\widehat{f} = f \circ \mathcal{A}^{\mathrm{t}}$ denotes its counterpart in the reference fluid domain. In particular, we have the following relation between current and reference fluid configuration:

$$\widehat{f}(\mathbf{x}_0, \mathrm{t}) = f(\mathcal{A}^{\mathrm{t}}(\mathbf{x}_0), \mathrm{t})\,, \qquad \mathbf{x}_0 \in \widehat{\Omega}_f\,. \tag{3.7}$$

In addition we define the ALE time derivatives to write the equations of fluid in the correct reference frame:

$$\left.\frac{\partial f}{\partial \mathrm{t}}\right|_{\widehat{\mathbf{x}}} : \Omega_{\mathrm{f}}^{\mathrm{t}} \times (0, \mathrm{T}) \to \mathbb{R}\,, \qquad \left.\frac{\partial f}{\partial \mathrm{t}}\right|_{\widehat{\mathbf{x}}}(\mathbf{x}, \mathrm{t}) = \frac{\partial \widehat{f}}{\partial \mathrm{t}} \circ (\mathcal{A}^{\mathrm{t}})^{-1}(\mathbf{x})\,, \qquad \mathbf{x} \in \Omega_f\,. \tag{3.8}$$

From the previous relation it is possible to calculate the fluid domain velocity as:

$$\mathbf{w}(\mathbf{x}, \mathrm{t}) = \left.\frac{\partial \mathbf{x}}{\partial \mathrm{t}}\right|_{\widehat{\mathbf{x}}} = \frac{\partial \mathcal{A}^{\mathrm{t}}}{\partial \mathrm{t}} \circ (\mathcal{A}^{\mathrm{t}})^{-1}(\mathbf{x})\,. \tag{3.9}$$

The nonlinear elastic models used for the structure have been described in chapter two. Regarding the fluid, the model used is the classical Newtonian incompressible fluid model.

In particular the governing equations are the Navier-Stokes equations, which in the Eulerian frame are the following:

$$\begin{cases} \rho_{\mathrm{f}} \dfrac{\partial \mathbf{u}}{\partial \mathrm{t}} + \rho_{\mathrm{f}}(\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} + \boldsymbol{\nabla} p - 2\boldsymbol{\nabla}\cdot(\mu_{\mathrm{f}}\mathbf{D}(\mathbf{u})) = \rho_{\mathrm{f}}\boldsymbol{f}_{\mathrm{f}}\,, \\ \nabla\cdot\mathbf{u} = 0\,, \end{cases} \tag{3.10}$$

in Ω_f^t and for $\mathrm{t} > 0$. They are supplemented by appropriate initial and boundary conditions which will be described later. It is necessary to rewrite them in the ALE formulation, as the computational domain Ω^{t} is moving. To do so, we apply the chain rule to the velocity time-derivative,

$$\left.\frac{\partial \mathbf{u}}{\partial \mathrm{t}}\right|_{\mathcal{A}} = \frac{\partial \mathbf{u}}{\partial \mathrm{t}} + \mathbf{w}\cdot\boldsymbol{\nabla}\mathbf{u}\,, \tag{3.11}$$

by which we get:

$$\begin{cases} \rho_{\mathrm{f}} \left.\dfrac{\partial \mathbf{u}}{\partial \mathrm{t}}\right|_{\mathcal{A}} + \rho_{\mathrm{f}}[(\mathbf{u}-\mathbf{w})\cdot\boldsymbol{\nabla}]\mathbf{u} + \boldsymbol{\nabla} p - 2\boldsymbol{\nabla}\cdot(\mu_{\mathrm{f}}\mathbf{D}(\mathbf{u})) = \rho_{\mathrm{f}}\boldsymbol{f}_{\mathrm{f}}\,, \\ \nabla\cdot\mathbf{u} = 0\,. \end{cases} \tag{3.12}$$

By coupling (3.12) to the equations of the structural problem described in Chapter 2, the FSI problem becomes the following:

Problem 3.1 (FSI problem)**.**

1. Fluid-structure problem. Find the velocity $\mathbf{u}$*, pressure* p *and the structure displacement* $\boldsymbol{\eta}$ *such that:*

$$\begin{cases} \rho_f \left.\dfrac{\partial \mathbf{u}}{\partial t}\right|_{\mathcal{A}} + \rho_{\mathrm{f}}\big[(\mathbf{u}-\mathbf{w})\cdot\boldsymbol{\nabla}\big]\mathbf{u} + \boldsymbol{\nabla} p - \boldsymbol{\nabla}\cdot\mathbf{T}_{\mathrm{f}} = \rho_{\mathrm{f}}\boldsymbol{f}_{\mathrm{f}}\,, & \text{in } \Omega_{\mathrm{f}}^t\times(0,T)\,, \\ \nabla\cdot\mathbf{u} = 0\,, & \text{in } \Omega_{\mathrm{f}}^t\times(0,T)\,, \\ \widehat{\rho}\,\dfrac{\partial^2\boldsymbol{\eta}}{\partial t^2} = \widehat{\rho}\,\mathbf{b} + \widehat{\nabla}\cdot\mathbf{P}\,, & \text{in } \widehat{\Omega}_s\times(0,T)\,, \\ \mathbf{u} = \dfrac{\partial\boldsymbol{\eta}}{\partial t}\circ(\mathcal{L}^t)^{-1}\,, & \text{on } \Sigma^t\times(0,T)\,, \\ \mathbf{T}_s\cdot\mathbf{n}_s + \mathbf{T}_{\mathrm{f}}\cdot\mathbf{n}_{\mathrm{f}} = 0\,, & \text{on } \Sigma^t\times(0,T)\,; \end{cases} \tag{3.13}$$

2. Geometry problem. Find the fluid domain displacement and velocity:

$$\begin{cases} \mathcal{A}^t(\widehat{\mathbf{x}}) = \widehat{\mathbf{x}} + Ext(\boldsymbol{\eta}(\widehat{\mathbf{x}}, t)|_{\widehat{\Sigma}}) \,, \\ \mathbf{w} = \dfrac{\partial \mathcal{A}^t}{\partial t} \circ (\mathcal{A}^t) \qquad in\ \widehat{\Omega}_f \times (0, \mathrm{T}) \end{cases} \tag{3.14}$$

The system is completed by suitable boundary conditions on $\partial\Omega$ and initial conditions. The fourth and fifth equations of system (3.13) are the coupling terms, and represent the continuity of fluid and structure velocities and the continuity of stresses. Often, in the discrete setting those conditions are decoupled, giving rise to the so-called partitioned (staggered) schemes. However, for the problem at hand it has been shown that staggered procedures are often unstable [15]-[40].

3.2 Weak formulation of the FSI problem

To derive the weak formulation (also called variational formulation) of the fluid-structure interaction problem (3.13)-(3.14) it is necessary to define the following functional spaces:

$$\begin{cases} \mathrm{V}_\mathrm{f}^\mathrm{t} = \{\mathbf{v}_\mathrm{f} \in \mathrm{H}^1(\Omega_\mathrm{f}^\mathrm{t})^\mathrm{d}\} \,, \\ \mathrm{Q}^\mathrm{t} = \mathrm{L}^2(\Omega^\mathrm{t}) \,, \\ \widehat{\mathrm{V}}_\mathrm{s} = \{\mathbf{v}_\mathrm{s} \in \mathrm{H}^1(\widehat{\Omega}_\mathrm{s})^\mathrm{d}\} \,, \\ \mathrm{S}^\mathrm{t} = \{(\mathbf{v}_\mathrm{f}, \mathbf{v}_\mathrm{s}) \in \mathrm{V}_\mathrm{f}^\mathrm{t} \times \widehat{\mathrm{V}}_\mathrm{s} : \mathbf{v}_\mathrm{f}\,|_{\Sigma^\mathrm{t}} = \mathbf{v}_\mathrm{s}\,|_{\widehat{\Sigma}} \circ \mathcal{L}^{-1}\} \,, \end{cases}$$

where L^p are Banach spaces and H^q are Sobolev spaces defined in appendix (A.1) and d is the space dimension (d=2 for bidimensional problem; d=3 for three-dimensional problem). Furthermore to obtain the weak formulation of the FSI problem, it is necessary to rewrite the convective and diffusive terms of the momentum equation of the fluid problem:

$$\begin{aligned} &a(\mathbf{u}, \mathbf{v}_\mathrm{f}) = \mu_\mathrm{f}\big(\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^\mathrm{T} : \boldsymbol{\nabla}\mathbf{v}\big) \,, \\ &c(\mathbf{u} - \mathbf{w}, \mathbf{u}, \mathbf{v}_\mathrm{f}) = \textstyle\int_{\Omega_\mathrm{f}^t} \big[(\mathbf{u} - \mathbf{w}) \cdot \boldsymbol{\nabla}\mathbf{u}\big] \cdot \mathbf{v}_\mathrm{f} \, \mathrm{d}\Omega \,, \end{aligned} \tag{3.15}$$

where $(\cdot, \cdot)$ is the inner product in $L^2(\Omega_f^t)$ defined in (A.1).
Using the spaces introduced and (3.15) the weak formulation of FSI problem (3.13) reads:

Weak-formulation 3.1 (FSI problem)**.** *For each $t \in (0, T)$ find $(\mathbf{u}(t), \boldsymbol{\eta}(t)) \in \mathrm{S}^t$, $p(t) \in \mathrm{Q}^t$ such that:*

$$\begin{aligned} &\rho_\mathrm{f}\left(\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathcal{A}}, \mathbf{v}_\mathrm{f}\right)_{\Omega_\mathrm{f}^t} + a(\mathbf{u}, \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t} - (p, \boldsymbol{\nabla} \cdot \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t} + c(\mathbf{u} - \mathbf{w}, \mathbf{u}, \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t} + \\ &+\rho_\mathrm{s}\left(\frac{\partial^2 \boldsymbol{\eta}}{\partial t^2}, \mathbf{v}_\mathrm{s}\right)_{\widehat{\Omega}_\mathrm{s}} + (\mathbf{T}_\mathrm{s}, \boldsymbol{\nabla}\mathbf{v}_\mathrm{s})_{\widehat{\Omega}_s} + (\boldsymbol{\nabla} \cdot \mathbf{u}, q)_{\Omega_\mathrm{f}^t} = (\mathbf{f}_\mathrm{f}, \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t} + (\mathbf{h}, \mathbf{v}_\mathrm{f})_{\Sigma_N^t} + (\mathbf{f}_\mathrm{s}, \mathbf{v}_\mathrm{s})_{\widehat{\Omega}_\mathrm{s}} \,, \end{aligned} \tag{3.16}$$

$$\mathbf{u}\,|_{\Sigma^t} = \left.\frac{\partial \boldsymbol{\eta}}{\partial t}\right|_{\widehat{\Sigma}} \circ \mathcal{L}^{-1}\,, \tag{3.17}$$

for any $(\mathbf{v}_\mathrm{f}, \mathbf{v}_\mathrm{s}) \in \mathrm{S}^t$ *and for any* $q \in \mathrm{Q}^t$.

The continuity condition on the velocity is imposed in the strong form while the stress continuity condition is imposed weakly. Moreover, the stress at the fluid interface can be interpreted as the residual of the weak formulation of the momentum equation considering a test function different from zero on Σ^t:

$$\begin{aligned}
&(\mathbf{T}_\mathrm{f}\cdot\mathbf{n}_\mathrm{f}, \mathbf{v}_\mathrm{f})_{\Sigma^\mathrm{t}} = \rho_\mathrm{f}\left(\left.\frac{\partial \mathbf{u}}{\partial \mathrm{t}}\right|_{\mathcal{A}}, \mathbf{v}_\mathrm{f}\right)_{\Omega_\mathrm{f}^\mathrm{t}} + a(\mathbf{u}, \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^\mathrm{t}} + c(\mathbf{u}-\mathbf{w}, \mathbf{u}, \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t} \\
&-(p, \boldsymbol{\nabla}\mathbf{v}_\mathrm{f}) - (\mathbf{f}_\mathrm{f}, \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t} - (\mathbf{h}, \mathbf{v}_\mathrm{f})_{\Sigma_N^\mathrm{t}} = -(\mathrm{R}_\mathrm{f}(\mathbf{u}, p), \mathbf{v}_\mathrm{f})_{\Omega_\mathrm{f}^t}
\end{aligned}$$

And similarly, so that, we have:

$$(\mathbf{T}_\mathrm{f}\cdot\mathbf{n}_\mathrm{f}, \mathbf{v}_\mathrm{f})_{\Sigma^t} + (\mathbf{T}_\mathrm{s}\cdot\mathbf{n}_\mathrm{s}, \mathbf{v}_\mathrm{s})_{\Sigma^t} = 0 \qquad \forall (\mathbf{v}_\mathrm{f}, \mathbf{v}_\mathrm{s}) \in \mathrm{S}^t\,.$$

Finally it is necessary to obtain the weak formulation of the geometry problem (3.14). To derive the last one it is necessary to introduce the following functional spaces:

$$\begin{aligned}
&\mathrm{M} = \{\boldsymbol{\psi} \in \mathrm{H}^1(\widehat{\Omega}) : \boldsymbol{\psi}\cdot\mathbf{n}_\mathrm{f}|_{\Sigma^\mathrm{t}} = 0\} \\
&\mathrm{M}^0 = \{\boldsymbol{\psi} \in \mathrm{M} : \boldsymbol{\psi}|_{\Sigma^\mathrm{t}} = 0\}
\end{aligned} \tag{3.18}$$

Consequently the weak formulation of (3.14) reads:

Weak-formulation 3.2 (Geometrical problem)**.** *Find* $\mathbf{w} \in \mathrm{M}$ *such that:*

$$\begin{cases}
(\boldsymbol{\nabla}\mathbf{w}, \boldsymbol{\nabla}\boldsymbol{\psi})_{\widehat{\Omega}_f} = 0\,, & \forall \boldsymbol{\psi} \in \mathrm{M}^0\,, \\
\mathbf{w} = \mathbf{u}\circ\mathcal{A}^t\,, & \text{on } \widehat{\Sigma}\,.
\end{cases} \tag{3.19}$$

3.3 Coupling strategies for the FSI problem

Before performing the time discretization of the system (3.13)-(3.14) we give an overview of the possible procedures to solve it. The fluid-structure interaction problem is defined by fluid equations, structure equations and by transmission conditions at the interface Σ, which we recall here:

$$\begin{cases}
\mathbf{u} = \dot{\boldsymbol{\eta}}\,, \\
\mathbf{T}_\mathrm{f}\cdot\mathbf{n}_\mathrm{f} + \mathbf{T}_\mathrm{s}\cdot\mathbf{n}_\mathrm{s} = 0\,.
\end{cases} \tag{3.20}$$

There are several ways to solve the FSI problem in practice. A monolithic approach is followed when fluid and structure are solved simultaneously in a single solver. Instead, a *partitioned* (*modular* or *segregated*) approach involves the use of possible pre-existing fluid and structure solvers which are then coupled throug (3.20). Furthermore, partitioned techniques

may be explicit or implicit, the latter requiring sub-iterations among fluid and structure solvers. With a monolitic approach it is easier to guarantee the stability in the energy norm of the discrete problem but we require a specific solver for a problem of a large size. The advantages of the partitioned procedures is that one can use a state-of-art method developed for the fluid and structure subproblems independently. In this thesis we are interested on *partitioned* algorithms. In an explicit partitioned algorithm, the fluid and structure sub-problem are solved once (or just few times) per time step. This approach is typical in aeroelastic problems, however is rather inconvenient in hemodynamics. In fact it can be shown that an explicit approach to FSI problem with significant added-mass effect (like haemodynamic problems where the structure and fluid densities are similar) is instable.

Stable partitioned schemes can be obtained by treating the interface conditions. This leads to the need of subiterating among fluid and structure and geometry solvers.

An additional possibility, called *semi-implicit*, is to treat the position of the interface and the convective term explicitly by the extrapolation of information from the previous time-steps, while conditions at the interface between fluid and structure are treated implicitly. At each time step the iterations between the two sub-problems are required. In this case there is a perfect balance of energy between fluid and structure problems, so the numerical schemes that are obtained with this approach are stable. However, the computational cost for problems with large added-mass effect, is very high because of the large number of sub-iterations needed at each time step.

Another possibility is to solve the monolithic, fully implicit problem by some iterative methods that iterate between fluid and structure subproblems. In this work the latter approach is adopted. It should also be noted that in addition to the inherent nonlinearity of the fluid-structure issue, adding the nonlinear stiffness term in the structure sub-problem.

3.3.1 Time discretization of the FSI problem

The time discretization of the system (3.13)-(3.14) requires the introduction of the fluid mesh displacement $\mathbf{d}_\mathrm{f}$ and the interface displacement $\mathbf{d}_\Sigma$. We introduce also the fluid solver operator $\mathcal{F}$, the solid solver operator $\mathcal{S}$, the mesh solver operator $\mathcal{M}$ and the coupling conditions operators, $\mathcal{C}_\mathrm{sf}$ and $\mathcal{C}_\mathrm{sm}$. Moreover, for simplicity, we use the convention: $g^p = g(\mathrm{t}^p)$. As shown in [41], a general implicit time discretization reads:

Algorithm 3.1 (Implicit time-discretization)**.** *Solve the nonlinear system of algebraic equations:*

$$\begin{cases} \mathcal{F}(\mathbf{u}^{n+1}, p^{n+1}, \mathbf{d}_\mathrm{f}^{n+1}) = \mathbf{f}_\mathrm{f}^{n+1}\,, \\ \mathcal{S}(\boldsymbol{\eta}^{n+1}) = \mathbf{f}_\mathrm{s}^{n+1}\,, \\ \mathcal{M}(\mathbf{d}_\mathrm{f}^{n+1}) = 0\,, \\ \mathcal{C}_\mathrm{sf}(\boldsymbol{\eta}_\Sigma^{n+1}, \mathbf{u}_\Sigma^{n+1}, p^{n+1}) = 0\,, \\ \mathcal{C}_\mathrm{sm}(\boldsymbol{\eta}_\Sigma^{n+1}, \mathbf{d}_{\mathrm{f},\Sigma}^{n+1}) = 0\,. \end{cases} \tag{3.21}$$

The dependence of the fluid solver operator on the movement of the fluid mesh should be clear as the fluid equations are solved in an unknown domain $\Omega_{\rm f}^{n+1} = \widehat{\Omega}_{\rm f} + \mathbf{d}_{\rm f}^{n+1}$.
It is possible to write the equations associated to each sub-problem as follows.

- $\mathcal{C}_{\rm sf}$: the coupling equations between solid and fluid, reads:

$$\begin{cases} \mathbf{u}_{\Sigma}^{n+1} = \dfrac{\boldsymbol{\eta}_{\Sigma}^{n+1} - \boldsymbol{\eta}_{\Sigma}^{n}}{\delta \mathrm{t}}\,, \\ \mathbf{T}_{\rm f} \cdot \mathbf{n}_{\rm f} + \mathbf{T}_{\rm s} \cdot \mathbf{n}_{\rm s} = 0\,. \end{cases} \tag{3.22}$$

- $\mathcal{C}_{\rm sm}$: the coupling equations between solid and mesh movement, reads:

$$\mathbf{d}_{\rm f,\Sigma}^{n+1} = \boldsymbol{\eta}_{\Sigma}^{n+1} \tag{3.23}$$

- $\mathcal{F}$: the fluid equations can be written using for instance the implicit-Euler algorithm as:

$$\begin{cases} \rho_{\rm f} \dfrac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\delta \mathrm{t}} + \rho_{\rm f}(\mathbf{u}^{n+1} - \mathbf{w}^{n+1}) \cdot \boldsymbol{\nabla} \mathbf{u}^{n+1} - \boldsymbol{\nabla} \cdot \mathbf{T}_{\rm f}^{n+1} = \mathbf{f}_{\rm f}^{n+1} & \text{in } \Omega_{\rm f}^{n+1}\,, \\ \boldsymbol{\nabla} \cdot \mathbf{u}^{n+1} = 0 & \text{in } \Omega_{\rm f}^{n+1}\,. \end{cases} \tag{3.24}$$

  where the fluid domain $\Omega_{\rm f}$ is an unknown of the problem.

- $\mathcal{S}$: the solid equations can be written using e.g. the Newmark method described in Chapter 2 as:

$$\begin{cases} \rho_{\rm s} \dfrac{\dot{\boldsymbol{\eta}}^{n+1} - \dot{\boldsymbol{\eta}}^{n}}{\delta t} - \boldsymbol{\nabla} \cdot \mathbf{P}^{n+1} = \mathbf{f}_{\rm s} & \text{in } \widehat{\Omega}_{\rm s}\,, \\ \dfrac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\delta \mathrm{t}} = \dfrac{\zeta}{2\theta} \dot{\boldsymbol{\eta}}^{n+1} + \Big(1 - \dfrac{\zeta}{2\theta}\Big) \dot{\boldsymbol{\eta}}^{n} - \dfrac{\delta \mathrm{t} \zeta}{2\theta} \Big[1 - \Big(1 + \dfrac{1-\zeta}{\zeta}\Big)\theta\Big] \ddot{\boldsymbol{\eta}}^{n} & \text{in } \widehat{\Omega}_{\rm s}\,. \end{cases} \tag{3.25}$$

- $\mathcal{M}$: the mesh equations can be written, for example, using the harmonic extension operator "Ext" as:

$$\begin{cases} \mathcal{A}^{n+1}(\widehat{\mathbf{x}}) = \widehat{\mathbf{x}} + \mathrm{Ext}(\boldsymbol{\eta}^{n+1} \mid_{\widehat{\Sigma}})\,, \\ \mathbf{w}^{n+1} = \dfrac{(\mathcal{A}^{n+1} - \mathcal{A}^{n}) \circ (\mathcal{A}^{n+1})^{-1}}{\delta \mathrm{t}}, \quad \Omega_{\rm f}^{n+1} = \mathcal{A}^{n+1}(\widehat{\Omega}_{\rm f})\,. \end{cases} \tag{3.26}$$

There are many techniques to compute the solution of (3.21) as explained for example in [42]. Here we consider a partitioned procedure that involves the interface degrees of freedom. In particular we use the Dirichlet-Neumann (DN) partitioned algorithm where the fluid is solved with the Dirichlet condition imposed by the structure, while the structure is solved with Neumann-type condition given by the fluid equations. Another possibility is to solve the fully nonlinear coupled system as show e.g. in [43].

3.3.2 Dirichlet-Neumann partitioned procedures

The Dirichlet-Neumann decoupling strategy, allows us to solve the fluid equations with a fixed domain, imposed by structure. The algorithm is well described in [22]. Here we breifly explain this method:

ALE map : Given the interface displacement $\boldsymbol{\eta}_\Sigma$, the computed ALE map and its velocity on the whole fluid domain is indicated by:

$$(\mathcal{A}^{k+1}, \mathbf{w}^{k+1}) = \mathcal{D}(\boldsymbol{\eta}_\Sigma^{k+1})\,; \tag{3.27}$$

Fluid : The Navier-Stokes equations in ALE formulation are solved on the new domain and pressure and velocity fields are obtained. We indicate this step as:

$$(\mathbf{u}^{k+1}, p^{k+1}) = \mathcal{F}(\mathcal{A}^{k+1}, \mathbf{w}^{k+1})\,; \tag{3.28}$$

Solid : The structure equations are solved using the Neumann condition imposed by fluid. In particular, the solid displacement and velocity are obtained:

$$(\dot{\boldsymbol{\eta}}^{k+1}, \boldsymbol{\eta}^{k+1}) = \mathcal{S}(\mathbf{u}^{k+1}, p^{k+1})\,. \tag{3.29}$$

$\mathcal{F}$ is, in our case, the Navier-Stokes solution operator, $\mathcal{S}$ is the structure solution operator previously introduced and $\mathcal{D}$ is the ALE map solution operator, in our case the harmonic extension operator.
It is possible to compose the operator to obtain an operator on the interface displacement:

$$\mathcal{T} = \gamma_\Sigma \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{D}\,, \tag{3.30}$$

where γ_Σ is the operator that maps the solid displacement and velocity into interface displacement:

$$\gamma_\Sigma : (\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) \to \boldsymbol{\eta}_\Sigma\,.$$

Finally the fixed-point problem on the interface displacement is:

$$\boldsymbol{\eta}_\Sigma^{(k+1)} = \mathcal{T}(\boldsymbol{\eta}_\Sigma^{(k)})\,, \tag{3.31}$$

which looks for a $\boldsymbol{\eta}_\Sigma$ such that:

$$\boldsymbol{\eta}_\Sigma = \mathcal{T}(\boldsymbol{\eta}_\Sigma) \tag{3.32}$$

3.3.3 Interface Newton-Krylov method

The fixed point problem (3.32) can be solved with more efficient iterative methods that (3.31). In this work we have used a Newton-Krylov method as proposed in [19] and [22] reformulating the problem in the following manner: at each time-step, find $\boldsymbol{\eta}_\Sigma$ such that

$$\mathcal{R}(\boldsymbol{\eta}_\Sigma) = \mathcal{T}(\boldsymbol{\eta}_\Sigma) - \boldsymbol{\eta}_\Sigma = 0\,. \tag{3.33}$$

Without going into details, which may be found in the cited literature, the Newton algorithm applied to the nonlinear system (3.33) shall perform the following steps [19]:

1. Choose an initial guess for structure interface displacement $\overline{\boldsymbol{\eta}}_{\Sigma}^{(0)}$.
2. Do until convergence:
   (a) Solve the Harmonic extension problem.
   (b) Evaluate the fluid solution operator $\overline{\mathbf{u}}_{\mathrm{f}}^{(k+1)} = \mathcal{F}(\overline{\boldsymbol{\eta}}_{\Sigma}^{(k)})$.
   (c) Evaluate the solid solution operator $\boldsymbol{\eta}_{\Sigma}^{(k+1)} = \mathcal{S}(\overline{\mathbf{u}}_{\mathrm{f}}^{(k+1)})$.
   (d) Evaluate the residual of the solid displacement $\mathcal{R}(\overline{\boldsymbol{\eta}}_{\Sigma}^{(k)}) = \boldsymbol{\eta}_{\Sigma}^{(k+1)} - \overline{\boldsymbol{\eta}}_{\Sigma}^{(k)}$.
   (e) Solve the tangent problem $\big[\mathrm{D}_{\boldsymbol{\eta}}\mathcal{R}(\overline{\boldsymbol{\eta}}_{\Sigma}^{(k)})\big]\delta\boldsymbol{\eta}_{\Sigma}^{(k)} = -\mathcal{R}(\overline{\boldsymbol{\eta}}_{\Sigma}^{(k)})$.
   (f) Update solid displacement: $\overline{\boldsymbol{\eta}}_{\Sigma}^{(k+1)} = \overline{\boldsymbol{\eta}}_{\Sigma}^{(k)} + \delta\boldsymbol{\eta}_{\Sigma}^{(k)}$,

where $\mathbf{u}_{\mathrm{f}} = (\mathbf{u}, \mathrm{p}, \mathbf{d}_{\mathrm{f}})$ and $\mathbf{d}_{\mathrm{f}}$ is the fluid displacement field. In particular, step 2-(e) is obtained by a matrix free GMRES Krylov iterative solver [44]. Hence, it only requires the evaluation of the operator $[\mathrm{D}_{\boldsymbol{\eta}}\mathcal{R}(\overline{\boldsymbol{\eta}})]$ applied to a solid state perturbations $\mathbf{z}$:

$$\big[\mathrm{D}_{\boldsymbol{\eta}}\mathcal{R}(\overline{\boldsymbol{\eta}})\big]\mathbf{z} = \big[\mathrm{D}_{\eta}\mathcal{T}\big]\mathbf{z} - \mathbf{z} = \Big(\gamma_{\Sigma} \circ \big[\mathrm{D}_{(\mathbf{u},p)}\mathcal{S}\big] \circ \big[\mathrm{D}_{(\mathcal{A},\mathbf{w})}\mathcal{F}\big] \circ \big[\mathrm{D}_{\boldsymbol{\eta}}\mathcal{D}\big]\Big)\mathbf{z} - \mathbf{z}\,. \tag{3.34}$$

The linear operator $\mathrm{D}_{\boldsymbol{\eta}}\mathcal{R}(\overline{\boldsymbol{\eta}})$ can be an exact or an inexact Jacobian. In the first case we speak of exact-Newton methods and the shape derivatives are used to calculate the perturbations of the fluid variables with respect to the ALE map. In the second case, the matrix of the tangent problem is approximated and we speak of quasi-Newton method. In this work we have adopted the latter strategy and we have neglected the shape derivatives.

The steps described above are defined in the *exactJacobianBase* template (EJ). After choosing the initial guess of the interface structure displacement $\overline{\boldsymbol{\eta}}_{\Sigma}^{(0)}$, the template *nonLinRichardson* is called, which recalls the method evalResidual into the template *exactJacobianBase*. Method evalResidual recalls the method eval that solves the harmonic extension, fluid and solid subproblem using the pre-existing solvers *HarmonicExtension*, *Oseen* and *NonLinearStructureSolver*. In particular, the fluid solver *Oseen*, solves the fluid problem using the Oseen approximation, hence the convective term is treated in a semi-implicit manner, while the structural solver *NonLinearStructureSolver* uses the Newton algorithm to solve the nonlinear part of the stiffness term as explained in Chapter 2.
Then the solid interface displacement residual is evaluated and compared with the prescribed tolerance:

$$\mathrm{toll} = \epsilon_{\mathrm{abs}} + \epsilon_{\mathrm{rel}}\big\|\mathcal{R}(\overline{\boldsymbol{\eta}}_{\Sigma}^{(0)})\big\|_{\mathrm{L}^{\infty}(\widehat{\Omega})} \tag{3.35}$$

If the interface solid displacement residual is less than the tolerance we have $\mathbf{u}_{\mathrm{f}}^{n+1} = \mathbf{u}_{\mathrm{f}}^{(k+1)}$ and $\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^{(k+1)}$ else, return to step 2-(a).

In particular NLSS perform all Newton iterations until convergence before moving to the tangent FSI problem solved with the GMRES method. This is a critical point, because we had to extend our structural solver NLSS, to correctly solve the tangent FSI problem previously

described. In particular, NLSS performs all Newton iterations until convergence, then the Jacobian matrix at the last Newton iteration is passed to FSI solver. To implement correctly the last point we have added two methods in NLSS, called iterateLin and applyBoundaryConditionsLin. The first one takes the structural Jacobian matrix when Newton method has converged, the second one applies the boundary conditions to the linearized problem. The schematic of the integration of NLSS into EJ is presented in the figure 3.3.3.

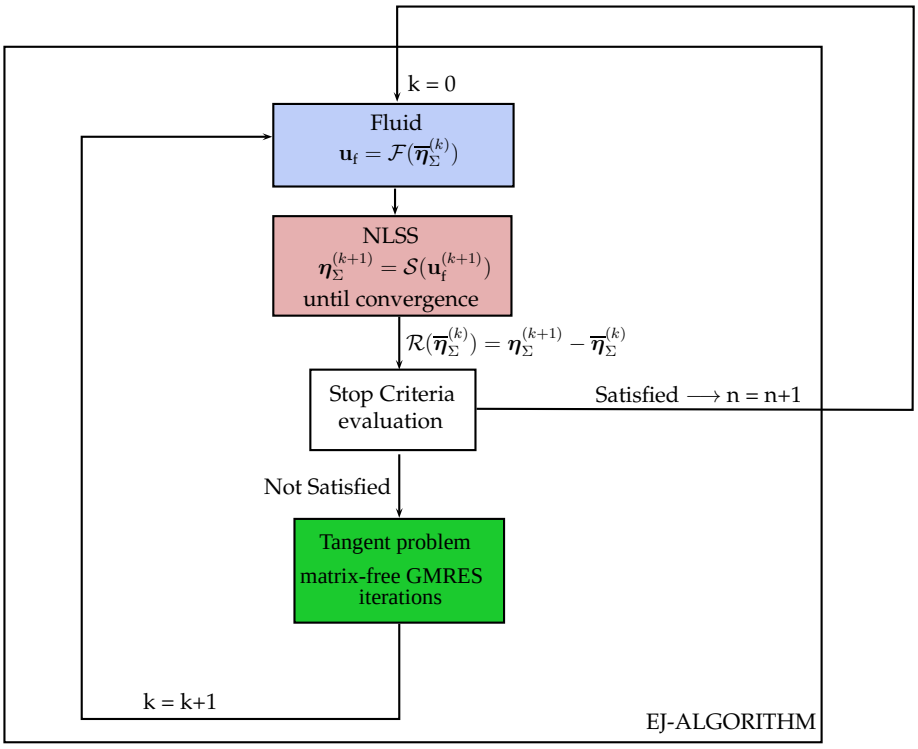


Figure 3.2: Integration of Nonlinear structural solver into FSI solver using Newton method

3.4 The design of the FSI solver

In the library LifeV-parallel, there were already two approaches to the solution of FSI problem for linear structure. The first is that using the fixed point method (3.31) and the corresponding solver is called *fixedPointBase*, while the second involves the use of Newton's method (3.33) and the corresponding solver is called *exactJacobianBase*. As remarked in the paragraph 3.3, we have used the second one.
The two solvers are supported by a complex structure of classes that containing the methods for the correct formulation of the problem and in particular we have the following architecture: The red circles in figure 3.3 indicate the main parts that we have modified and integrated into FSI solver. In particular the principal modifications concern the structural part of the solver with the addition of the *NonLinearStructureSolver*. Other manipulations of

Figure 3.3: FSI solver: general view

the FSI solver are needed also in *FSIOperator*, *FSISolver*, *dataFSI* and *exactJacobianBase*. Moreover, we emphasize the parts modified to obtain the FSI solver with nonlinear hyperelastic structural models. We note, however, that we have tried to build a structural solver compatible with the existing FSI solvers. In the following list, we give a brief description of each functional block, represented above, with two exception: *fixedPointBase* and *exactJacobianBase*. In particular, for the FSI problem we have used the second algorithm, *exactJacobianBase* which is described in the dedicated paragraph 3.4.3.

- *FSISolver*: is a class that contains four main methods, setData(const data_PtrType& data), setup(), initialize() and iterate(). The first one, recalls the data of FSI problem from data file. Setup() initializes the FSI problem, while initialize() gives the initial condition for time scheme. Finally, iterate(), is the temporal loop.
  Here we have modified just one method, initialize() to recall the correct method of *NonLinearStructureSolver*.

- *FSIOperator*: is a class that contains the definition of the solvers for Interface problem, fluid subproblem and structure subproblem. Moreover, it contains some methods to provide the correct description of the problem. The main methods are partitionMeshes() to manage the solid and fluid part of the mesh, setupFEspace() to manage the finite-element space, setupDOF() to manage the degrees of freedom of the FSI problem,

setupIDOF() to manage the interface degrees of freedom, setupFluidSolid() to recall the harmonic extension, fluid and solid constructor and to initialize the members of fluid and solid subproblems, buildSystem() to compute the constant matrices, updateSystem() to update the non-constant matrices at each time-step. *FSIOperator* contains many others methods to treat the conditions on interface and to set some parameters useful as stop criteria that are not described because they are not the goal of this work.
Here we have modified some methods. First one is setupFluidSolid() to call the correct constructor of the nonlinear structural solver. Moreover, buildSystem() and updateSystem() are modified and now recalls also the methods of the *NonLinearStructureSolver*.

- *DataFSI*: is a class that contains some methods to manage the data of FSI problem. *DataFSI* is connect to two other classes to manage structure and fluid data: *dataElasticStructure* and *dataOseen*.

- *VenantKirchhofSolver*: is a factory that provides some functions to characterize the problem of elasticity. *VenantKirchhofSolver* has a derived class called *LinearVenantKirchhofSolver* to solve the linear elasticity problem.
We have modified this point with the addition of the *NonLinearStructureSolver* as a derived class of *VenantKirchhofSolver*.

- *OseenSolver*: is a class to solve the fluid problem using the Oseen equations [45]. In particular, the Oseen equations in the primitive variables $\mathbf{u}$ and p, are linear due to the approximation of the convective term as: $\rho(\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} \approx \rho(\mathbf{U}\cdot\boldsymbol{\nabla})\mathbf{u}$, where $\mathbf{U}$ is the steady fluid velocity and $\mathbf{u}$ are the perturbations of steady fluid velocity.

- *HarmonicExtensionProblem*: is a class to solve the elliptic problem of the harmonic extension.

- *main*: is the script that recalls the principal methods of the classes decribed above. This script is connected with *BoundaryConditions* and *ud_functions* to manage the boundary conditions of FSI problem.

3.4.1 *FSISolver* and *FSIOperator*'s principal methods

In this brief paragraph we show the principal methods of *FSIOperator* and *FSISolver* that represent the core of FSI solver. The principal methods *exactJacobianBase* are explained in the subsections 3.4.3. We want to remark that in the base classes of the FSI solver, such as *FSISolver* and *FSIOperator*, we have not modified several methods, because we have adapted the *NonLinearStructureSolver* to the FSI solver in an iterative process as specified previously.

FSISolver's principal methods

The principal method of *FSISolver* (FSIS) is iterate, which recalls the template NLR. Other important methods are also setData, setup and Initialize. In the following list we describe the tasks of each method.

- setData: Manages MPI to parallel computing, define the data file and calls the method of *FSIOperator* setData.

- setup: Calls three method of *FSIOperator*: setupFluidSolid, setupSystem and buildSystem. In particular we have modified two of these three methods: setupFluidSolid and buildSystem that now calls also the class related to the NLSS.

- Initialize: Initializes the temporal schemes. This method has been modified, because in the previous solver the time scheme is a mid-point method that does not require the initial data for the acceleration.

- iterate: Calls the method updateSystem of *FSIOperator* and calls the template NLR to solve the nonlinear fluid-structure problem. Here we have not modified anything because we have created the compatible methods in the NLSS.

FSIOperator's principal methods

FSIOperator (FSIO) implements a variety of methods to manage the boundary condition at the interface, to compute and update the matrices of fluid and structure and many others method. Principal methods are buildSystem, updateSystem, couplingVaribleExtrap and some methods to setting up the data file, the FE space and other quantities. In the following list we describe briefly the tasks of the main method of FSIO and the modifications we have made.

- setDataFile: Initializes data using GetPot [46].

- setupFESpace: Setup the finite-element spaces. In particular defines the FE order and the quadrature rules.

- setupDof: Identifies the degrees of freedom and related maps.

- createInterfaceMaps: Creates the interface maps and in particular the solid and fluid variables.

- setupIDof: Identifies the interface degree of freedom.

- setupFluidSolid: Sets up the Harmonic extension, fluid and solid subproblems. In particular the constructors of each subproblem and their setup methods are recalled. Here, we have modified in particular the call of the constructor of the solid subproblem. Now this method calls in a correct way the constructor of the *NonLinearStructureSolver*.

- setupSystem: Reads the data files with the information of each subproblem.

- buildSystem: Calls buildSystem method of fluid and solid solvers that computes the constant matrices.

- updateSystem: Updates the Harmonic extension, fluid and solid subproblems. In particular updates the right-hand side of each subproblem. Moreover calls shiftSolution and couplingVariableExtrap methods.

- shiftSolution: Updates the right-hand side of fluid subproblem.

- couplingVariableExtrap: Updates the interface displacement and velocity have to be passed to the next time-step.

- initializeFluid: Gives the initial conditions for mesh and fluid subproblems calling the specific methods of each solvers.

- initializeSolid: Gives the initial conditions for solid subproblem calling the specific method of NLSS.

- moveMesh: Moves the mesh and recomputes the fluid matrices.

- transferFluidOnInterface: Transfers the fluid quantities on interface.

- transferSolidOnFluid: Transfers the solid quantities on fluid.

- trasferSolidOnInterface: Transfers the solid quantities on interface.

- trasferInterfaceOnSolid: Transfers the interface quantities on interface.

3.4.2 FSI solver architecture

As previously done for the structural solver, we show in the figure 3.4 the interaction between blocks to give a schematic view on how the FSI solver works.

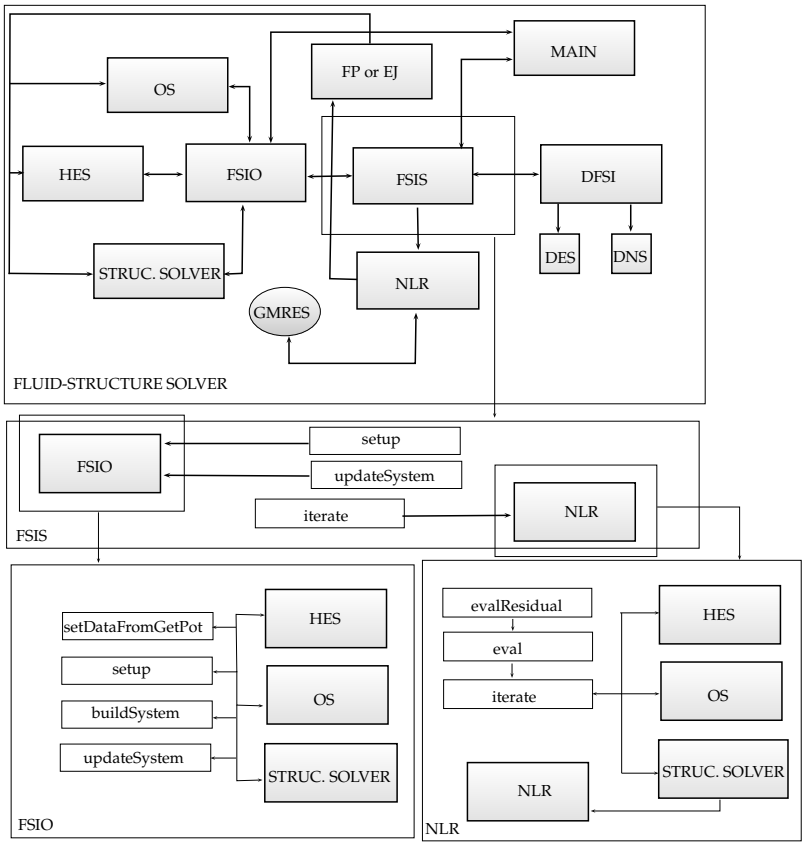


Figure 3.4: FSI solver design

Principal methods of *exactJacobianBase*

To perform the steps described above there are some methods defined into the class *exactJacobianBase*. We are interested to show how it is possible to complete a time-step. Hence the main methods of *exactJacobianBase* are evalResidual, eval, solveJac, solveLinearFluid and solveLinearSolid which are also the methods modified to a correct integration of the NLSS into the FSI solver. In the following list we give a short description of each method:

- evalResidual: Evaluates the stopping criterion on solid displacement residual. In particular here we have not done modifications. The stopping criterion is on interface displacement between the actual iteration and the previous iteration.

- eval: Solves geometrical, fluid and solid subproblems, recalling the principal methods of each solver. In particular recalls the iterate method for *meshMotion*, *fluid* and *solid* subproblems. Here we have correctly interfaced the NLSS, recalling the methods of the nonlinear structural solver.

- solveJac: Solves the linearized FSI problem.

- Apply: Calls solveLinearFluid and solveLinearSolid and manages the transmission conditions.
- solveLinearFluid: Solves the fluid linear problem used for transmission conditions subiterations.
- solveLinearSolid: Solves the solid linear problem used for transmission conditions subiterations. In particular, we have created a specific method for this class to solve the linearized problem of the structure. Here we pass the Jacobian matrix obtained from the Newton method on the solid subproblem when it arrives to convergence.

Architecture of *exactJacobianBase*

As done for the overall FSI solver, here we detail the architecture of the *exactJacobianBase* template.

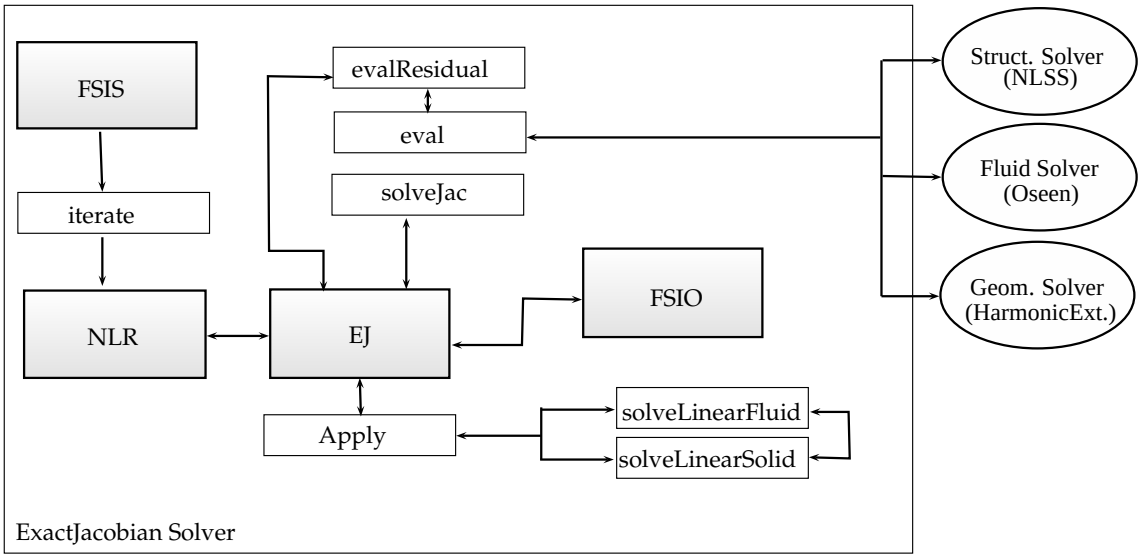


Figure 3.5: EJ architecture

3.5 Test case on a cylindrical straight vessel

In this section we present the results obtained from 4 simulations, using a structured mesh for fluid and structure. The radius of the straight cylinder in undeformed configuration is R_0 = 0.5 cm with a length of L = 5 cm. Structure's thickness is h = 0.1 cm. Data for fluid and structure are reported in the table below:

Table 3.1: Structural and Fluid data set for cylindrical straight vessel test

μ_f[poise]	ρ_f[g/cm^3]	ρ_s[g/cm^3]	E[dyne/cm^2]	ν	γ	κ[dyne/cm^2]	α[dyne/cm^2]
0.03	1.0	1.2	6×10^6	0.45	0.80	1×10^8	2×10^6

The boundary conditions applied to the problem are the following:

- Structure basis: embedded;
- Structure outer surface: Stress free;
- Structure inner surface: Neumann condition from the fluid subproblem;
- Fluid inlet second case: $p = 1.332e5\sin(\frac{\pi t}{0.003})$ until t < 0.003s. $p = 0$ for t > 0.003s ;
- Fluid outlet: Adbsorbing boundary condition;
- Fluid inner surface: Dirichlet condition from the structure subproblem.

The space and time discretization parameters are in the table 3.2. The mesh used is sparse

Table 3.2: Space and time discretization parameters

t_0[s]	t_N[s]	δt[s]	Displacement FE	Pressure FE	Velocity FE
0	0.01	0.0001	P1	P1	P1-Bubble

and its data are reported in table 3.3. Finite-element used are P1 for fluid pressure, P1 bubble for fluid velocity, and P1 for structure displacement.

Table 3.3: Mesh properties of solid subproblem (*vessel20*) and fluid subproblem (*tube20*)

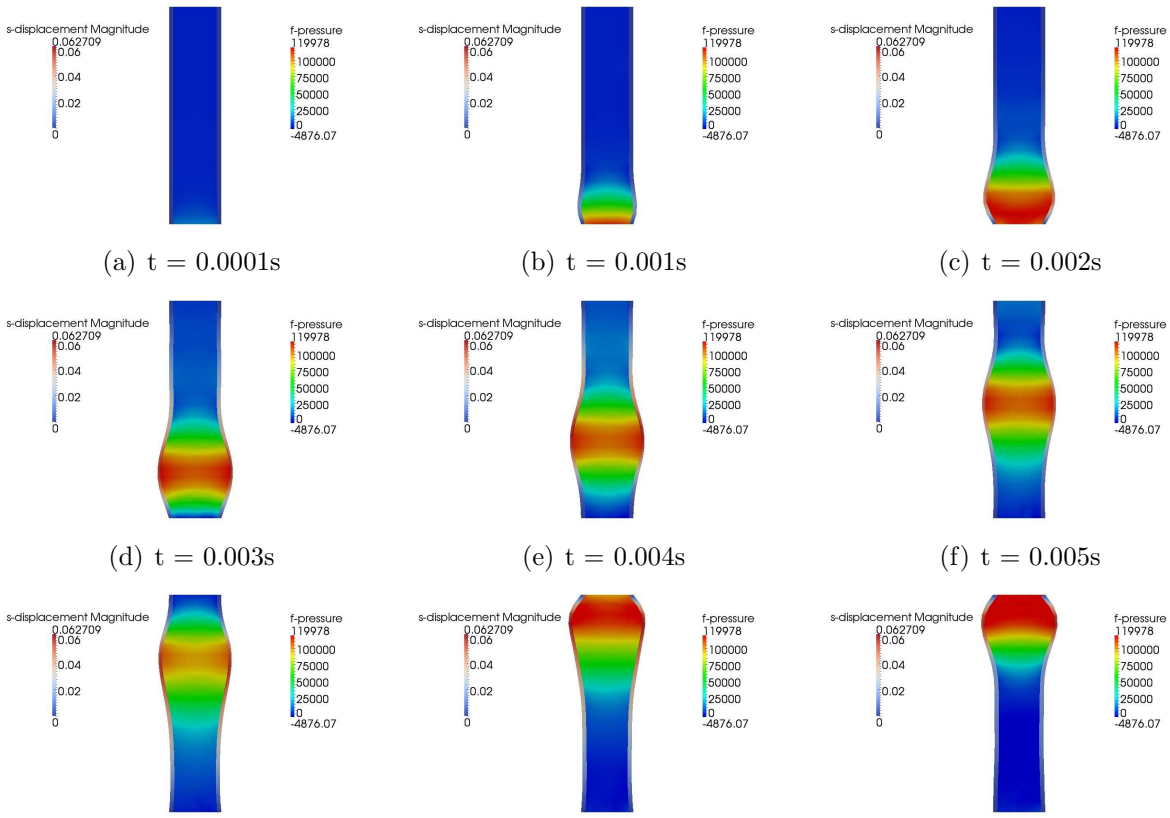
	Nodes	**Triangles**	**Thetrahedra**	**Length**	**Inner radius**	**Outer radius**
vessel20	1360	1760	4800	5 cm	0.5 cm	0.6 cm
tube20	1050	956	4680	5 cm	–	0.5 cm

It is possible to see in figure 3.6 a qualitative picture of the simulation. In particular two different time-steps are shown and it is possible to note the pressure wave propagation in the vessel.

(a) t = 0.003s (b) t = 0.006s

Figure 3.6: Qualitative picture of the FSI test on straight cylinder

Figures, 3.7-3.8-3.9-3.10, indicate the fluid pressure in the plane section of the vessel and the solid displacement. In particular we want to show the pressure wave propagation. Hence the values of pressure and displacement are fixed at t = 0.003 s and are not rescaled at each time-step.
Figures 3.11(a)-3.12(a)-3.13(a)-3.14(a), indicate the fluid velocity in the plane section of the vessel. In this case we are interested in the modulus of the velocity field and data are rescaled at each time-step.
From these figures it is already possible to note some qualitative properties. In particular we note that the linear material presents less stiffness with respect to the other 3 nonlinear materials. In fact, if we see the maximum magnitude of displacement of linear structure model, we note a value of $|\boldsymbol{\eta}_{\text{Lin}}| = 0.062709$ cm, which compared with the nonlinear models, $|\boldsymbol{\eta}_{\text{SVK}}| = 0.055542$ cm, $|\boldsymbol{\eta}_{\text{NH}}| = 0.052501$ cm, $|\boldsymbol{\eta}_{\text{Exp}}| = 0.053375$ cm, presents the highest value. In addition, the pressure wave propagation is faster for nonlinear materials, which are more rigid, in accord with the theory. Also, we can see that the speed in the planar section is lower for nonlinear models. From these preliminary considerations we can conclude that the results obtained are qualitatively reasonable and the FSI problem is solved correctly.

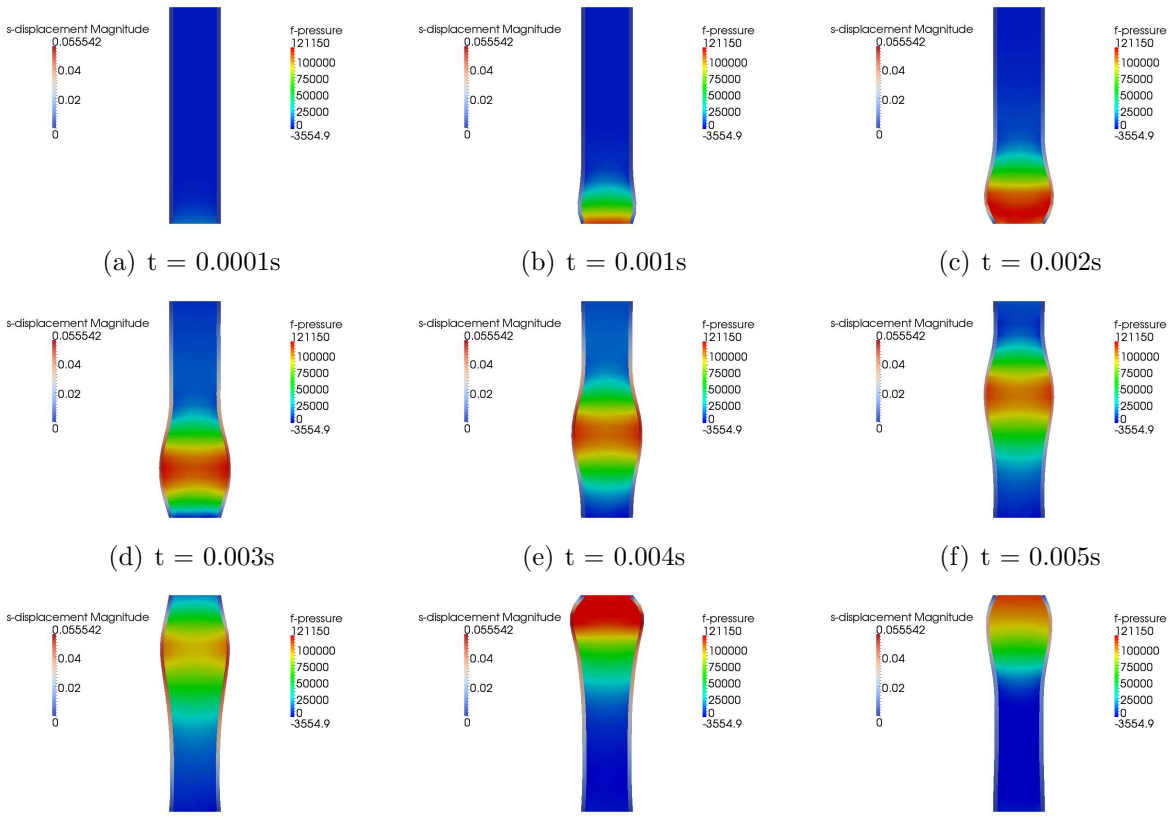


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.7: Planar section of the vessel for 9 different time-steps, linear structural model. Fluid pressure[dyne/cm^2]and solid displacement[cm] using 133322 dyne/cm^2 as inlet pressure

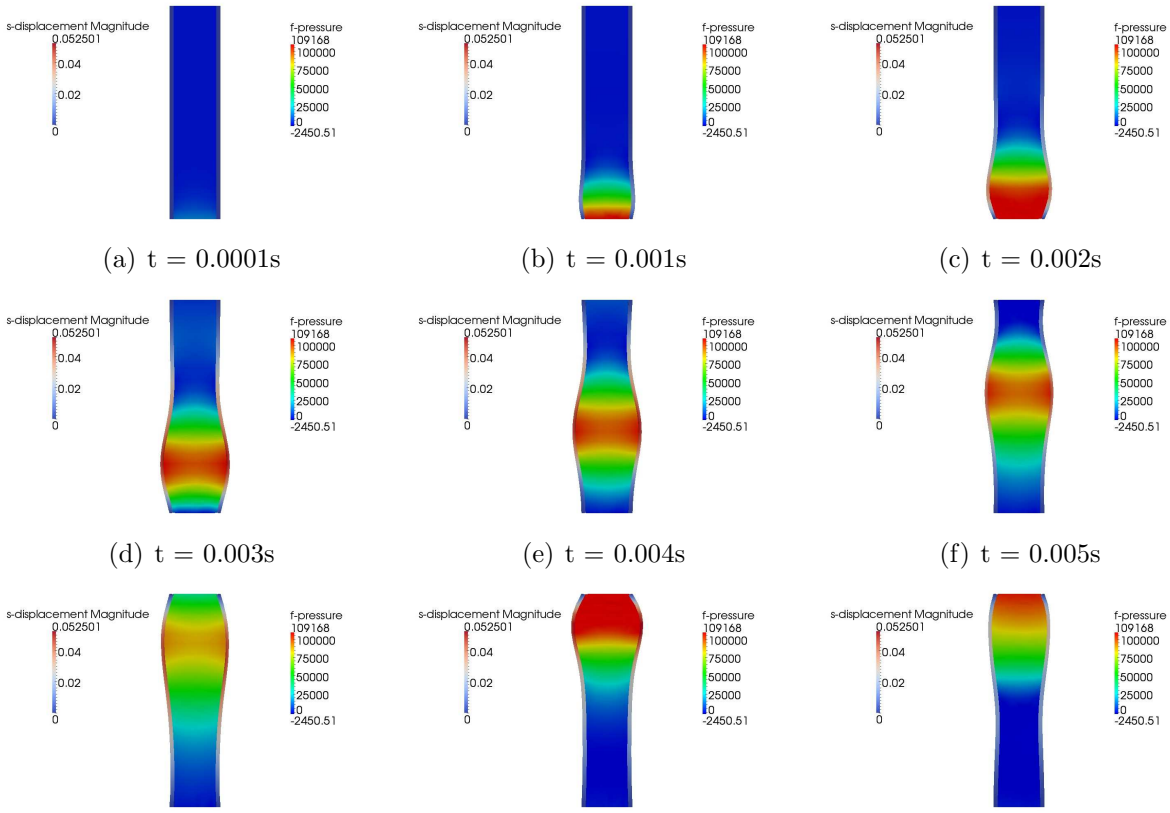


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.8: Planar section of the vessel for 9 different time-steps, St.Venant-Kirchhoff structural model. Fluid pressure[dyne/cm^2]and solid displacement[cm] using 133322 dyne/cm^2 as inlet pressure

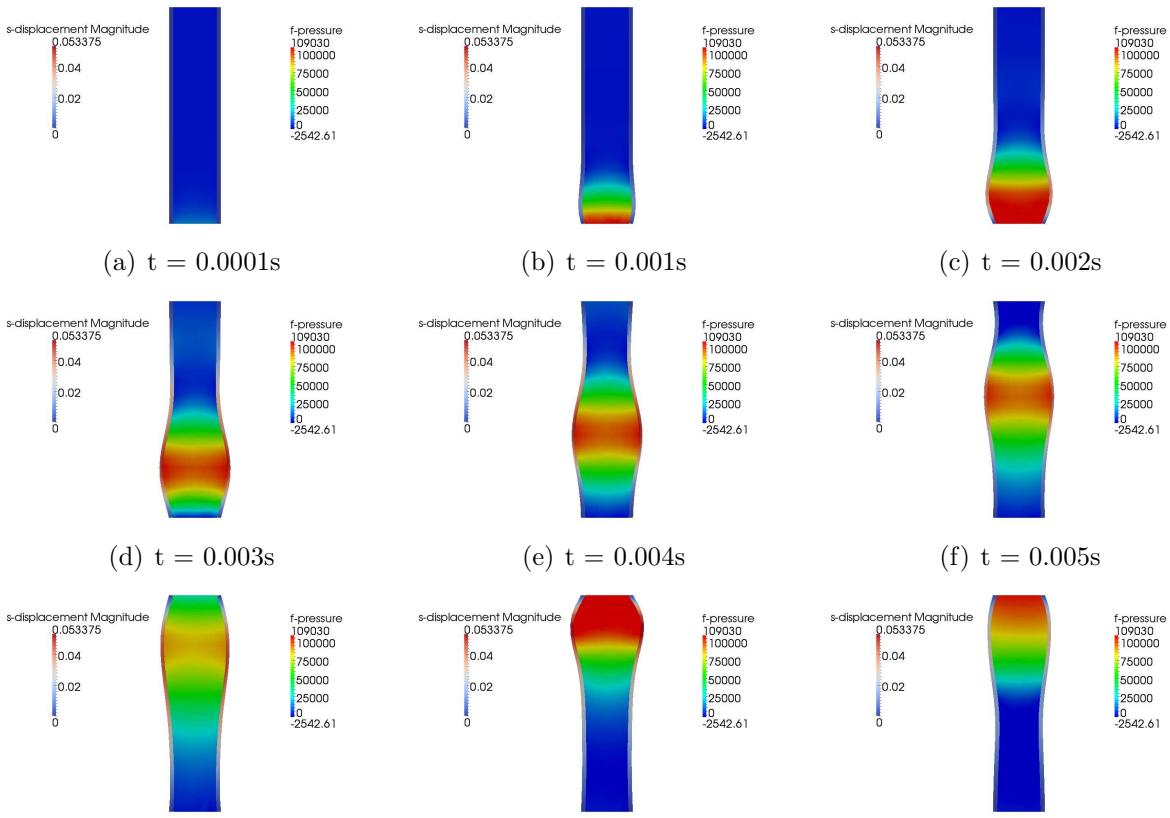


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.9: Planar section of the vessel for 9 different time-steps, Neo-Hookean structural model. Fluid pressure[dyne/cm^2]and solid displacement[cm] using 133322 dyne/cm^2 as inlet pressure

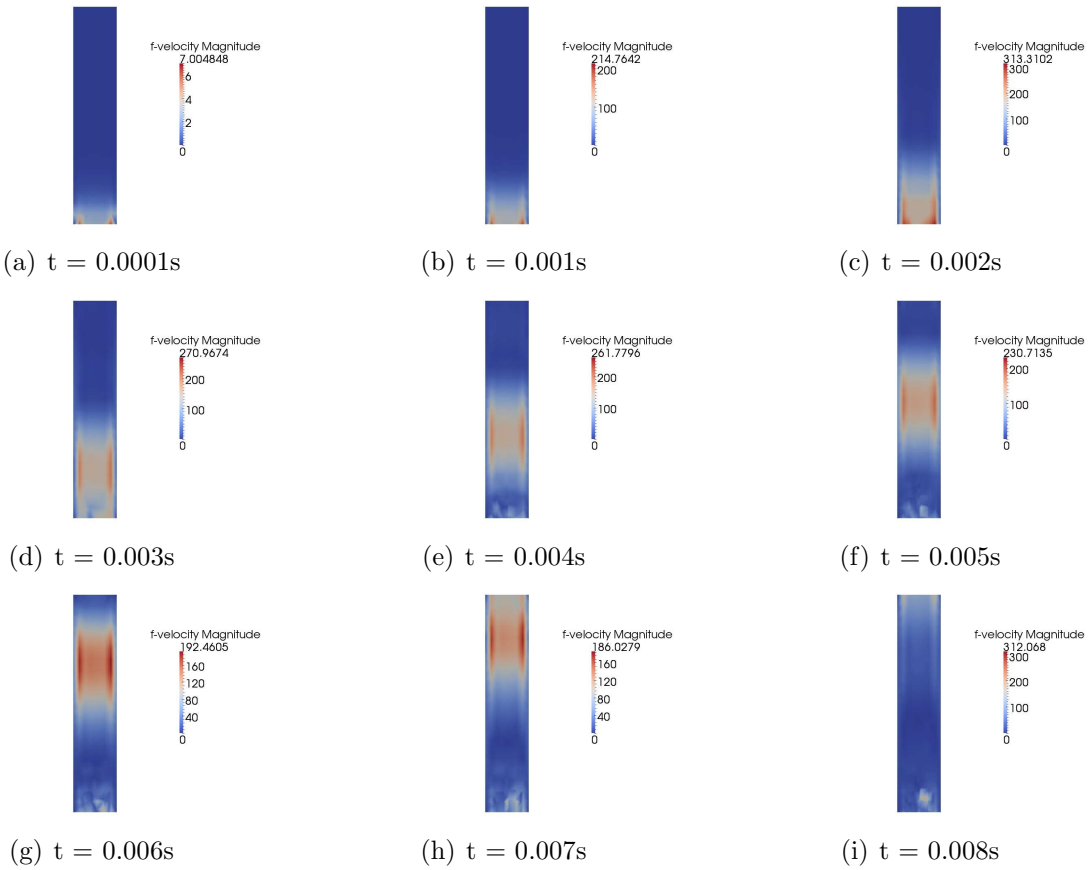


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.10: Planar section of the vessel for 9 different time-steps, Exponential structural model. Fluid pressure[dyne/cm^2]and solid displacement[cm] using 133322 dyne/cm^2 as inlet pressure

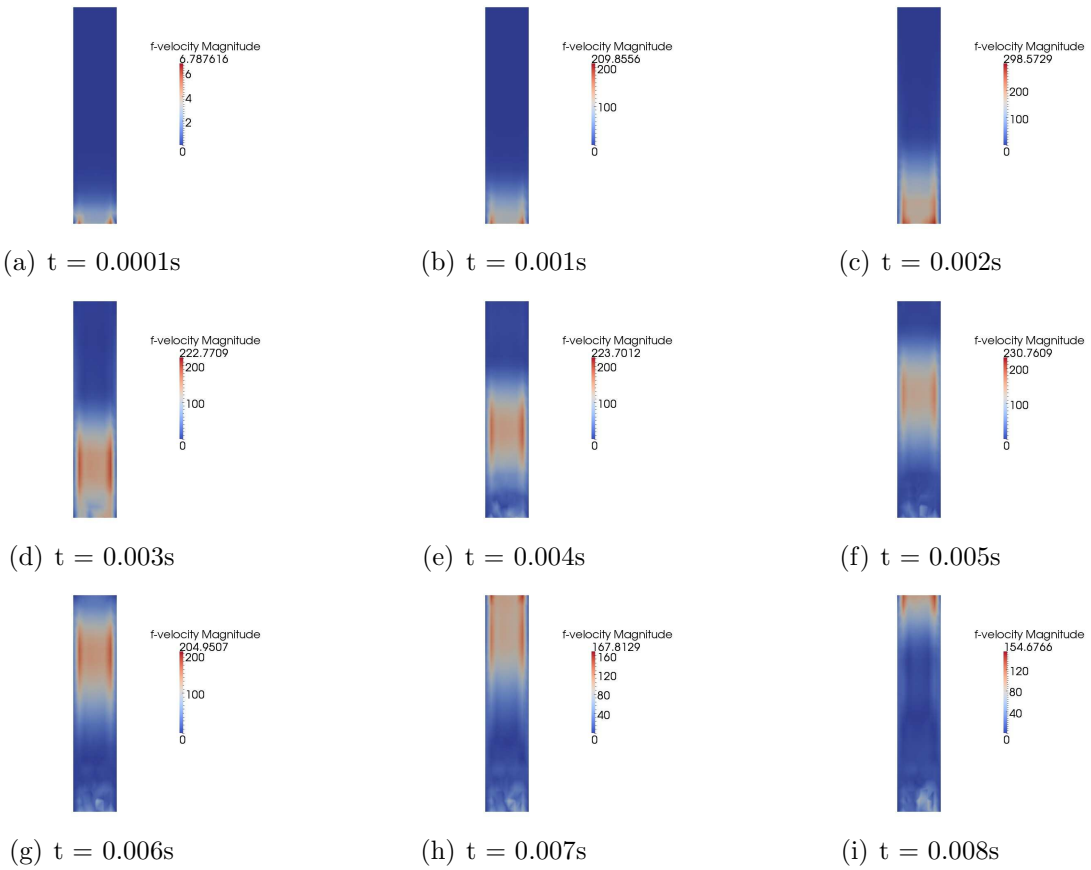


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.11: Planar section of the vessel for 9 different time-steps, Linear structural model. Fluid velocity[cm/s] using 133322 dyne/cm^2 as inlet pressure

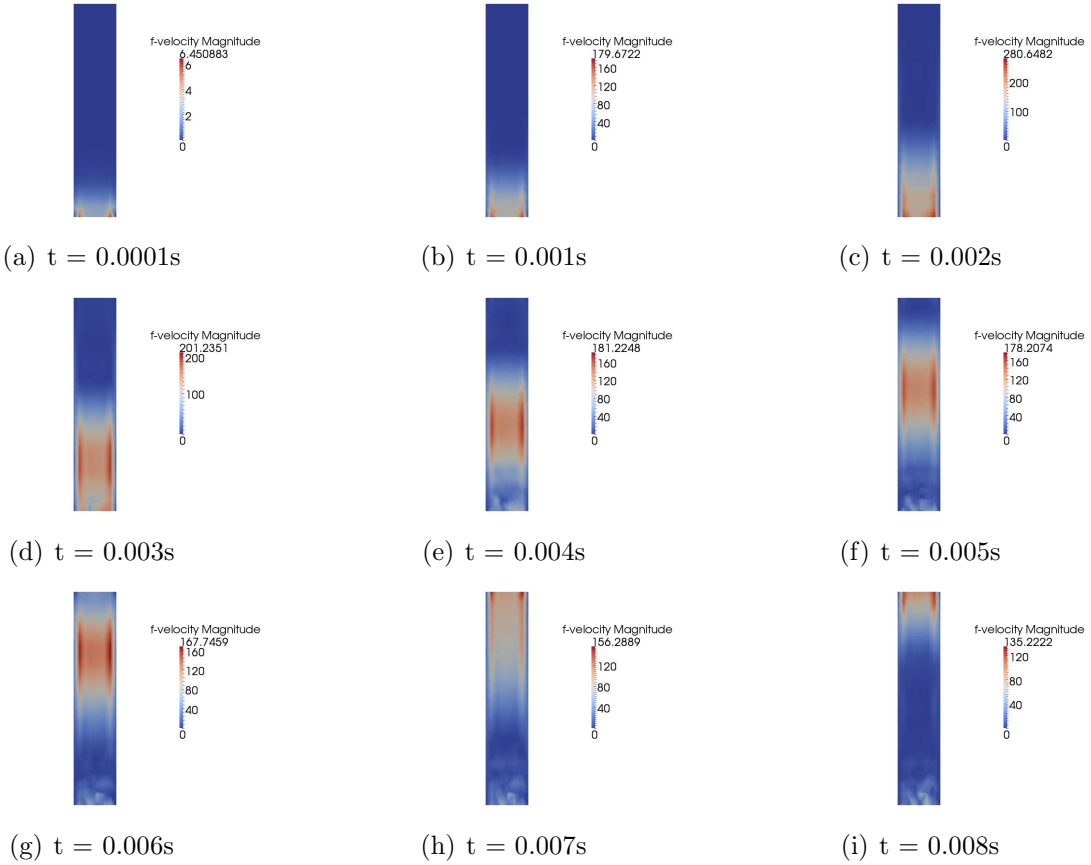


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.12: Planar section of the vessel for 9 different time-steps, St.Venant-Kirchhoff structural model. Fluid velocity[cm/s] using 133322 dyne/cm^2 as inlet pressure

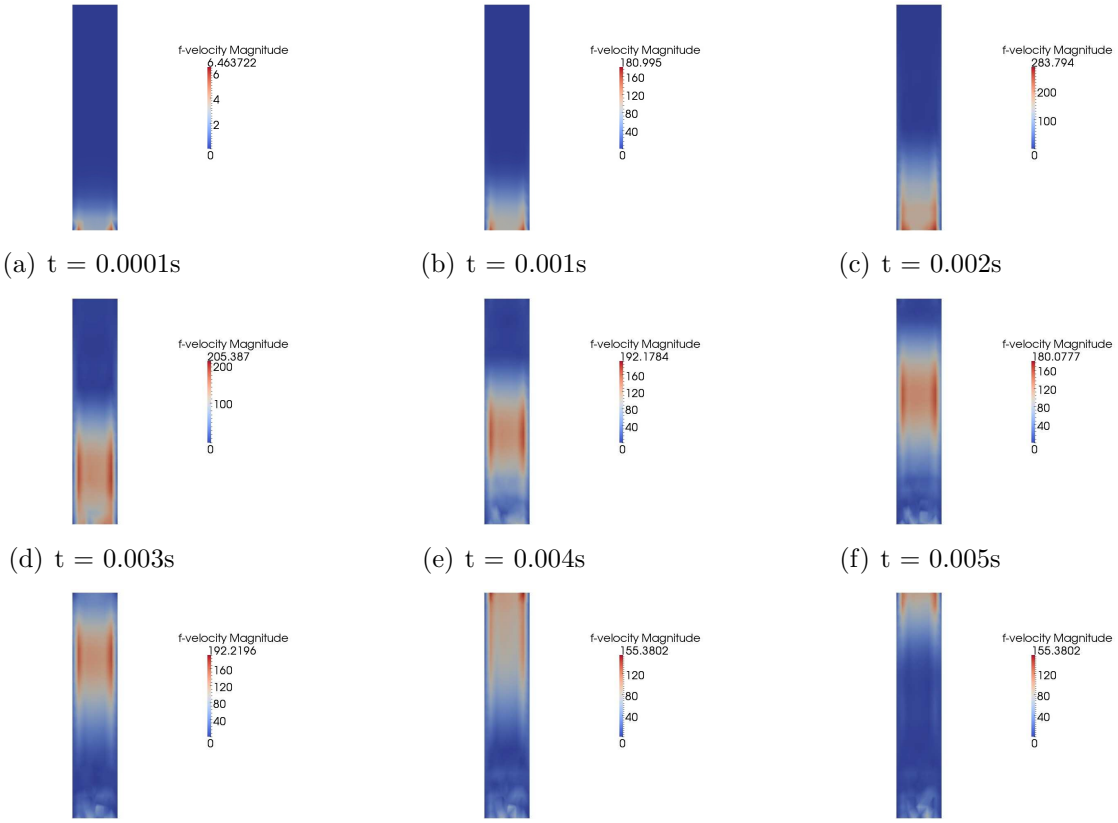


(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.13: Planar section of the vessel for 9 different time-steps, Neo-Hookean structural model. Fluid velocity[cm/s] using 133322 dyne/cm^2 as inlet pressure



(a) t = 0.0001s (b) t = 0.001s (c) t = 0.002s

(d) t = 0.003s (e) t = 0.004s (f) t = 0.005s

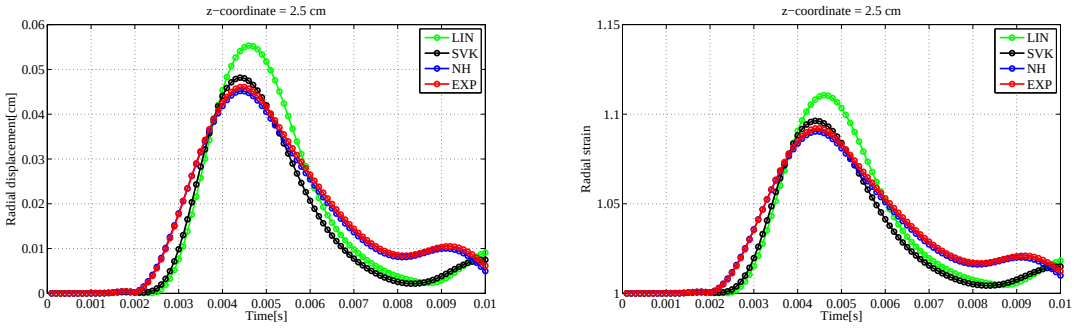
(g) t = 0.006s (h) t = 0.007s (i) t = 0.008s

Figure 3.14: Planar section of the vessel for 9 different time-steps, Exponential structural model. Fluid velocity[cm/s] using 133322 dyne/cm^2 as inlet pressure

Now, we present the most significant results obtained from the mean section of the cylinder (z-coordinate = 2.5 cm). In particular, a comparison between linear material and three nonlinear structural models is done in terms of primitive variables for fluid and structure.

In figure 3.16, the magnitude of displacement 3.16(a) and the magnitude of strain 3.16(b) vs. time of an internal node of the structure are shown. We note how the nonlinear structural models, for strain levels greater than 5%, show a higher stiffness than the linear model, as indeed we see in the figure 3.15 for the radial displacement 3.15(a) and strain 3.15(b).

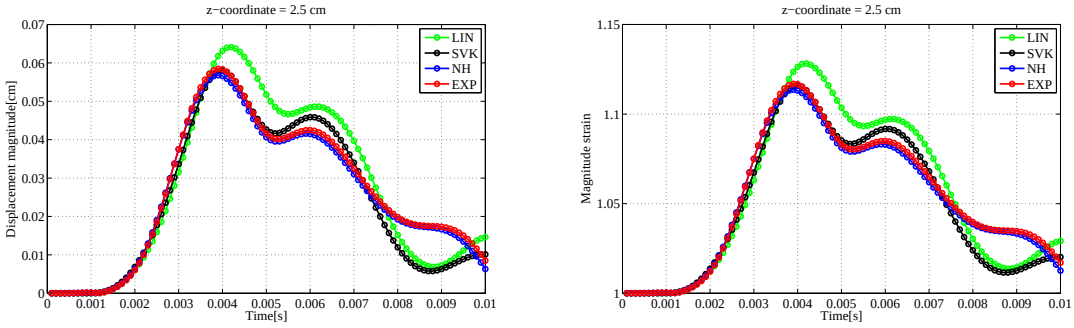
The curve of the mean axial velocity 3.17(a) and mean pressure 3.17(b) are very similar between linear elasticity and nonlinear structural models. Finally we have compared the mean radial fluid velocity 3.18(a) between 4 materials and the mean fluid pressure versus radial strain 3.18(b). We note how the Neo-Hookean and Exponential materials have higher peaks of radial velocity than linear elsticity and St.Venant-Kirchhoff models. Similarly, 3.18(b), confirms the result observed in 3.15, ie the nonlinear structural models are more rigid than linear model at high level of deformation.



(a) Radial displacement(cm) vs. time(s)

(b) Radial strain vs. time(s)

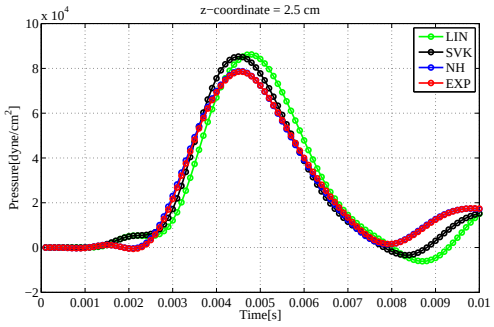
Figure 3.15: Comparison of solid radial displacement[cm] and strain vs. time[s] for linear material and 3 nonlinear structural models. FSI simulation: straight vessel



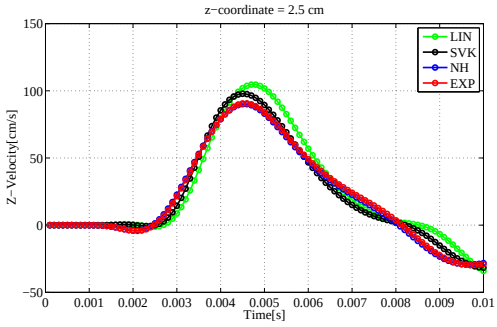
(a) Magnitude of displacement(cm) vs. time(s)

(b) Magnitude of strain vs. time(s)

Figure 3.16: Comparison of solid magnitude of displacement[cm] and strain vs. time[s] for linear material and 3 nonlinear structural models. FSI simulation: straight vessel

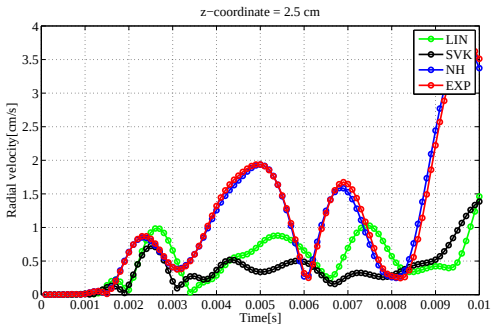


(a) Fluid pressure(dyne/cm^2) vs. time(s)

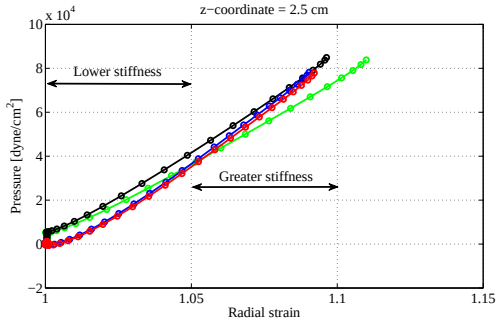


(b) Axial velocity (cm/s) vs. time(s)

Figure 3.17: Comparison of fluid pressure [dyne/cm^2] and axial velocity[cm/s] vs. time[s] for linear material and 3 nonlinear structural models. FSI simulation: straight vessel



(a) Radial fluid velocity(cm/s) vs. time(s)



(b) Fluid pressure (dyne/cm^2) vs. solid radial strain

Figure 3.18: Comparison of fluid radial velocity[cm/s] vs. time[s] and pressure-strain curves for linear material and 3 nonlinear structural models. FSI simulation: straight vessel

Chapter 4

Details of the implementation

In this chapter we give some more details on the implementation of the algorithm in the object-oriented open-source code LifeV. In particular we emphasize that the code works on parallel architecture and we remember that we indicate it by *LifeV-parallel*. Furthermore we explain the most important aspects of the structural solver implementation and its integration into FSI algorithm.

4.1 LifeV-parallel

As previously remarked, LifeV is a finite element (FE) library that provides the implementations of state of the art mathematical and numerical methods. It is a C++ object-oriented open-source code under the LGPL license (for more details see [47]) and it serves both as a research and production library. In particular it has been ported to GNU/Linux systems, standard UNIX systems, Cygwin systems and AIX based systems. LifeV is a joint collaboration among four institutions: École Polytechnique Fédérale de Lausanne (CMCS) in Switzerland, Politecnico di Milano (MOX) in Italy, INRIA (REO/ESTIME) in France and Emory University (Sc. Comp) in the U.S.A. The software covers a wide range of problems, such as fluid structure interaction, structural mechanics and fluid dynamics.

The parallel version of LifeV exists since 2006, it is based on the Trilinos packages [48] and can be used on a wide range of computers, ranging from PCs to large supercomputers, like the BlueGene/P, Cray XT5 and XE6. It contains around 50000 lines of C++ code and it is growing. It is a project within the SourceForge like environment GForge hosted on http://cmcsforge.epfl.ch providing bug tracking, task management, account management, secure connection and a Git repository (for more details see [49]).

Parallel computing requires more a complex programming effort with to serial computing. In fact, it is necessary to manage processes on multiple CPUs and handle their communication properly. In particular, processes or objects must be able to send and receive messages to other processes or objects. In LifeV-parallel this task relies on the standard Message Passing Interface (MPI), developed by the Message Passing Interface Forum (MPIF) [50]. The partition of the mesh is based on ParMETIS. ParMETIS is an MPI-based parallel library for

partitioning unstructured graphs, meshes and for computing fill-reducing ordering of sparse matrices. Moreover, as mentioned above, the parallel version of LifeV is based on Trilinos packages that provide some specified mathematical tools. The principal Trilinos' packages used in LifeV-parallel and relevant for our work are the following[1]:

- **Epetra**: It manages vectors and matrices and it is organized in classes. LifeV-parallel uses the primary parallel user classes, reported below:

  1. Communicator class: Epetra_Comm. Contains specific information about the parallel machine we are using. Currently supports serial (Epetra_SerialComm), MPI (Epetra_MpiComm) and prototype hybrid MPI/threaded parallel programming models.
  2. Map classes: Epetra_Map, Epetra_LocalMap, Epetra_BlockMap. Contain information used to distribute vectors, matrices and other objects on a parallel (or serial) machine.
  3. Vector class: Epetra_Vector. Real double precision vector class. Supports construction and use of vectors on a parallel machine.
  4. Multi-vector class: Epetra_MultiVector. Real double precision multi-vector class. Supports construction and use of multi-vectors on a parallel machine. A multi-vector is a collection vectors. It is a generalizaion of a 2D array.
  5. Sparse row graph class: Epetra_CrsGraph. Allows construction of a serial or parallel graph.
  6. Pure virtual row matrix class: Epetra_RowMatrix. Pure virtual class that specifies interfaces needed to do most of the common operations required by a row matrix. Any class that implements Epetra_RowMatrix can be used with Epetra_LinearProblem and AztecOO.
  7. Easier-to-implement row matrix class: Epetra_BasicRowMatrix. Epetra_RowMatrix has many pure virtual functions that must be implemented by an adaptor.
  8. Sparse row matrix class: Epetra_CrsMatrix. Real double precision sparse matrix class. Supports construction and use of row-wise sparse matrices. Epetra_FECrsMatrix is a specialization that supports finite element applications more naturally.
  9. Sparse block row matrix class: Epetra_VbrMatrix. Real double precision block sparse matrix class. Supports construction and use of row-wise block sparse matrices. Epetra_FEVbrMatrix is a specialization that supports finite element applications more naturally.
  10. Jagged diagonal sparse matrix class: Epetra_JadMatrix. Real double precision sparse matrix class for vector processors. Constructs and updates a jagged-diagonal format sparse matrix from an existing Epetra_RowMatrix.

[1]These short notes were taken from the site http://trilinos.sandia.gov and from the specific user-guide supplied by Sandia National Laboratory

11. Import/Export classes: Epetra_Import and Epetra_Export. Constructed from two Epetra_BlockMap (or Epetra_Map or Epetra_LocalMap). Allows efficient transfer of objects built using one map to a new object with a new map. Supports local and global permutations, overlapping Schwarz operations [51] and many other data movement algorithms.

- **Ifpack**: It provides a suite of object-oriented algebraic preconditioners for the solution of preconditioned iterative solvers. In fact, for the parallel solution of algebraic system, are often used Krylov type iterative solvers (see, for example [52]). It is well-known that the use of Krylov-type methods require good spectral properties of the iteration matrix of the system A. In fact, to obtain high performance on the convergence properties it is necessary that the matrix A is well conditioned. This rarely happens, in fact, for large algebraic systems, the matrix A is often ill-conditioned. In addition, the stiffness matrix resulting from the FEM discretization of the elasticity operator has a condition number that scales like h^{-2}, thus it degrades rapidly as $h \to 0$. Therefore it is necessary to formally manipulate the original system and solve a system of the form:

$$\mathrm{P}^{-1}\mathrm{A}\mathrm{M}^{-1}\mathbf{y} = \mathrm{P}^{-1}\mathbf{b}\,, \mathrm{M}\mathbf{x} = \mathbf{y}\,, \tag{4.1}$$

  where P and M are invertible matrices called left and right preconditioners, respectively. Often, M = I and only the left preconditioner is active. The preconditioning matrices have to satisfy the following requirements:

  1. They must be "easily invertible", that is the cost of solving $\mathrm{P}\mathbf{z} = \mathbf{r}$ and $\mathrm{M}\mathbf{x} = \mathbf{y}$ must be much smaller than the cost of solving of solving $\mathrm{A}\mathbf{x} = \mathbf{b}$ directly.
  2. $cond(\mathrm{P}^{-1}\mathrm{A}\mathrm{M}^{-1}) << cond(\mathrm{A})$, that is the spectrum of the preconditioned matrix $\mathrm{P}^{-1}\mathrm{A}\mathrm{M}^{-1}$ has to be more "clustered" around 1 than the original matrix.
  3. The condition number of the preconditioned matrix should be indipendent from the matrix size and, in a parallel setting from the number of processors employed.
  4. Clearly in a parallel setting, the cost of the application of the preconditioner should scale with the number of processors. Of course, it is extremely difficult to obtain all these requirements at the same time, the choice of a preconditioner is often a matter of compromise.

  Ifpack provides some single-level algebraic preconditioners for parallel large scale applications that can be classified as follows:

  1. *Relaxation schemes*: Jacobi, Gauss-Seidel, Symmetric Gauss-Seidel, etc.
  2. *Polynomial preconditioner*: Neumann, Least Square and Chebyshev.
  3. *Incomplete factorization preconditioner*: IC, ILU, ILUT, etc.
  4. *One-level domain decomposition preconditioners of Schwarz-type.*
  5. *Sparse Approximate Inverses*: SPAI, AINV, etc.

It is interesting to see the single-level preconditioner can be used in conjunction with multilevel preconditioner. See for example [53]. For the structural problem and for the fluid and geometrical problems, the preconditioner used is *One-level domain decomposition preconditioners of Schwarz-type* which unfortunately does not scale well with the number of subdomains. However it is possible to obtain scalability by adding a coarse operator. For more details on Ifpack package see [54].

- **ML**: It is a multigrid preconditioning package that solves linear systems of type A$\mathbf{x}$ = $\mathbf{b}$, where A is a generic sparse matrix, $\mathbf{b}$ is a known vector and $\mathbf{x}$ is the unknown. In particular, the basic idea of a multigrid solver consists to approximate the original problem on a hyerarchy of *grids* and use *solutions* from coarse grids to accelerate the convergence on the finest grid. For more details see [53]. Multigrid techniques have been originally developed as solvers, yet they have proved to be valid preconditioners.

- **AztecOO**: It is a collection of C++ classes that support the construction and use of objects for solving linear systems of equations of the form A$\mathbf{x}$ = $\mathbf{b}$, via preconditioned Krylov methods. For more details on AztecOO package see also [52].

- **Amesos**: It provides an object-oriented interface to several serial and parallel sparse direct solvers libraries for the solution of the linear system of equations A$\mathbf{x}$ = $\mathbf{b}$, where A is a real sparse, distributed matrix, defined as an Epetra RowMatrix object, and $\mathbf{b}$ are defined as Epetra MultiVector objects. In particular, Amesos makes the following steps to perform efficiently the solution of the linear system above:

  1. Definition of the sparsity pattern of the linear system matrix;
  2. Computation of the symbolic factorization;
  3. Definition of the values of the linear system matrix;
  4. Computation of the numeric factorization;
  5. Definition of the values of the right-hand side;
  6. Solution of the linear system.

  For more details on Amesos package see also [55].

For a complete list and description of Trilinos' packages, see [48].
y Finally, to solve unsymmetric sparse linear systems, A$\mathbf{x}$ = $\mathbf{b}$, **Umfpack** is used (see also [56]).

LifeV is made up of seven modules, divided according to the task they perform. In figure 4.1 they are represented with their functionality. In particular, the folder LIFECORE, contains the basic classes of LifeV, as for example Factory to manage the factories, or Chrono to manage the timer. Folder LIFEALG, contains the interfaces with linear solvers as AztecOO, Amesos and with the preconditioners as Ifpack and ML. Moreover it contains NonLinearRichardson template to linearize non linear system, and NonlinearLineSearch that implements

the line-search backtracking method. LIFEARRAY, contains some files for handling matrices and vectors, and the management of tensors. LIFEFEM has some methods to manage the Finite-elements as, for example, quadrature rules. Furthermore, it also contains the files for the management of boundary conditions and some temporal schemes templates as TimeAdvance. LIFEFILTERS contains the classes to manage the pre and post-processing. In particular two exporter are defined, ensight (a visualization package) and hdf5 (a tool for parallel i/o). LIFEMESH contains the classes to manage the mesh. Finally, LIFESOLVER contains the classes to solve some mathematical and physical problem. In particular in this folder there are the classes that defines mechanical problem, the classes that defines the fluid-structure interaction problem and many others.

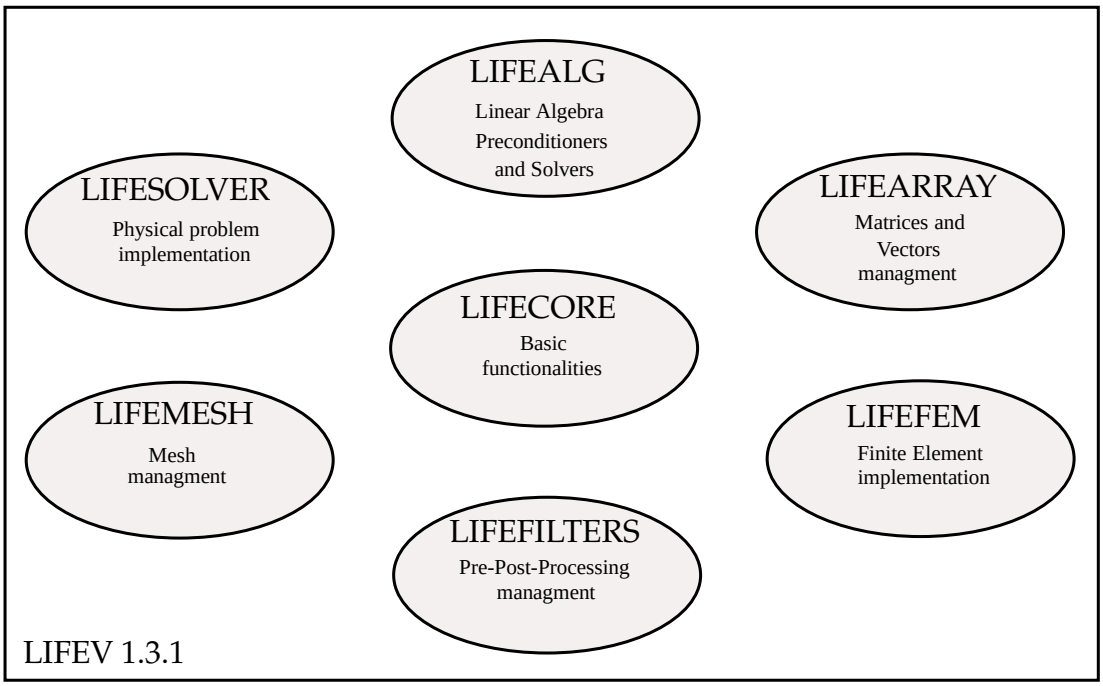


Figure 4.1: Principal folders of LifeV

4.2 Structural solver implementation and code performances

Having outlined in Chapter 2, the architecture of the structural solver, we detail now its implementation. In particular, we analyze the steps necessary to perform a time step. Hence the main methods of NLSS that we describe will be: setup, buildSystem, updateSystem, iterate, evalResidual, solveJacobian, updateJacobian. Moreover we show a brief overview on the constructor NonLinearStructureSolver and on the template nonLinRichardson used to linearize the nonlinear system. Finally we evaluate the code performances in terms of CPU-time to perform the main steps of the algorithm and the correct parallelization by a scalability test

on Lagrange cluster.

4.2.1 Methods description

In the following list, we show the main methods of NLSS and the template NLR.

NonLinearStructureSolver():

The guidelines in the development of LifeV involve the use of constructors without argument. Initially, we have used a nonlinear solver's constructor having as argument the finite element space. The reason for this, is due to the implementation of the Neo-Hookean and Exponential materials that currently require the use of local tensors. In fact, these local tensors do not provide a constructor with no argument, but must be initialized by defining the dimensions of each, thus necessitating the finite element space. To work around this problem we have used pointers to local tensors. Surely this aspect will be improved in the development of the solver. In particular the lines' code that implement the constructor and the local tensors' pointer are the following:

```
//! Definition of local tensors
...
typedef KNMK<Real>                    KNMK_Type;
typedef boost::shared_ptr<KNMK_Type> KNMKPtr_Type;

//! Definition of Constructor
NonLinearStructureSolver( );

//! Local tensors as pointers
...
KNMKPtr_Type M_Fk;

//! Constructor
template <typename Mesh, typename SolverType>
NonLinearStructureSolver<Mesh, SolverType>::
NonLinearStructureSolver( ):
    M_data                        ( ),
    M_FESpace                     ( ),
    ...
    ...
    M_First_PiolaK_Stress         ( ),
    M_elvecK                      ( ),
    M_VecK                        ( )
```

where `SolverType` is Trilinos package previously introduced.

Setup:

The setup will be reset all the objects defined in the constructor. There are three setup methods that differ from each other for arguments that are passed. One of this three setup is defined below:

```
template <typename Mesh, typename SolverType>
void NonLinearStructureSolver<Mesh, SolverType>::
setup( boost::shared_ptr<data_type>                          data,
       const boost::shared_ptr< FESpace<Mesh, EpetraMap> >& dFESpace,
       boost::shared_ptr<Epetra_Comm>&                       comm,
       const boost::shared_ptr<const EpetraMap>&             monolithicMap,
       UInt                                                  offset ){
M_data          = data;
M_FESpace       = dFESpace;
...
M_trCisok.reset ( new KN_Type( dFESpace->fe().nbQuadPt() ) );
M_trCk.reset    ( new KN_Type( dFESpace->fe().nbQuadPt() ) );}
```

It is interesting to note that in parallel version it is required a *communicator* between the processes, implemented in the fifth line of the code above boost::shared_ptr<Epetra_Comm>& comm.

buildSystem:

The method buildSystem computes the mass matrix of the system and the linear stiffness matrix. The construction of the mass matrix:

$$\mathrm{M}_{ij} = \frac{2}{\delta \mathrm{t}^2} \int_{\widehat{\Omega}_\mathrm{h}} \widehat{\rho}\, \phi_j \phi_i \, \mathrm{d}\widehat{\Omega} \tag{4.2}$$

is performed with the following lines of code and is done for all three nonlinear materials:

```
for ( UInt i = 1; i <= this->M_FESpace->mesh()->numVolumes(); i++ ){
      this->M_elmatM->zero();
      mass( dti2 * this->M_data->rho(), *this->M_elmatM,
      this->M_FESpace->fe(), 0, 0, nDimensions );

      for ( UInt ic = 0; ic < nc; ic++ ){
           assembleMatrix( *this->M_mass, *this->M_elmatM,
           this->M_FESpace->fe(), this->M_FESpace->dof(), ic, ic,
           this->M_offset +  ic*totalDof, this->M_offset +  ic*totalDof);}}
```

In particular in the third line, the local matrix `M_elmatM` is initialized to zero, in the fourth-fifth lines, method `mass`, defined into EO file, builds the mass term depending on the finite element used. Finally the lines 8-9-10 assemble the mass matrix into the global variable `M_mass` that corresponds to the matrix M_{ij} defined in 4.2.

The linear stiffness matrix is done only for St.Venant-Kirchhoff material and corresponds to the following terms of the first Piola-Kirchhoff stress tensor:

$$\mathbf{P}(\boldsymbol{\eta}) = \lambda(\mathrm{div}\boldsymbol{\eta})\mathbf{I} + \mu(\boldsymbol{\nabla}\boldsymbol{\eta} + \boldsymbol{\nabla}\boldsymbol{\eta}^{\mathrm{T}})\,. \tag{4.3}$$

Its implementation corresponds to the following lines of code:

```
for ( UInt i = 1; i <= this->M_FESpace->mesh()->numVolumes(); i++ ){
      this->M_elmatK->zero();
      switch (this->M_data->constitutive_law()){
          case 0:
               stiff_strain ( 2.0*mu, *this->M_elmatK, this->M_FESpace->fe() );
               stiff_div    ( lambda, *this->M_elmatK, this->M_FESpace->fe() );
          break;
          case 1:
          break;
  ...}
     for ( UInt ic = 0; ic < nc; ic++ ){
           for ( UInt jc = 0; jc < nc; jc++ ){
               assembleMatrix( *this->M_linearStiff, *this->M_elmatK,
               this->M_FESpace->fe(), this->M_FESpace->fe(),
               this->M_FESpace->dof(), this->M_FESpace->dof(), ic,
               jc, this->M_offset +ic*totalDof, this->M_offset + jc*totalDof );}}}
```

where in the third line, the local matrix `M_elmatK` is initialize to zero, while in the lines 7-8 two terms of equation 4.3 are computed and in the lines 22-23-24-25 they are assembled.

updateSystem:

The method `updateSystem` is called once per time step. This method calculates the right-hand side without boundary conditions. Also calculates the right-hand side of velocity and acceleration, which are useful in the update of the two. As already mentioned the method of time integration is Newmark. In particular, it called another method of NLSS, `updateNonLinearTerms` which calculates the nonlinear stiffness terms. The principal code's line of `updateSystem` are reported below:

```
...
updateNonlinearTerms(mat_tmp, vec_tmp);
...
mat_tmp->GlobalAssemble();
*mat_stiff  += *mat_tmp;
vec_tmp->GlobalAssemble();
*vec_stiff   = *vec_tmp;
...
*mat_stiff += *M_linearStiff;
mat_stiff->GlobalAssemble();
```

```
...
Real DeltaT     = M_data->dataTime()->getTimeStep();
vector_type M_z = *M_disp;
M_z            +=  DeltaT*(*M_vel);
...
*this->M_rhsNoBC  = *this->M_mass*M_z;
*this->M_rhsNoBC -= (*mat_stiff)*coef*(*this->M_disp);
*this->M_rhsNoBC -= (*vec_stiff)*coef;

*M_rhsA = (2.0 / (this->M_data->dataTime()->zeta() * pow(DeltaT,2))) * M_z +
          ((1.0 - this->M_data->dataTime()->zeta()) /
          (this->M_data->dataTime()->zeta())) * (*M_acc);

*this->M_rhsW = *this->M_vel + ( 1 - this->M_data->dataTime()->theta() )*
                 DeltaT * (*M_acc);
```

where, first line recalls method `updateNonLinearTerms` and computes nonlinear stiffness terms. Lines 4-5-6-7 assign the ausiliary variables to stiffness vector and stiffness matrix. The call `->GlobalAssemble();` is necessary to assembly correctly vectors and matrices when they are computed in a parallel way. Line 9, adds to the stiffness matrix the linear part of stiffness, computes into `buildSystem`. The last line of the code computes the right-hand side of the problem:

$$\mathsf{rhsNoBC} = \frac{2}{\delta \mathrm{t}^2}\mathbf{M}(\boldsymbol{\eta}^n + \delta \mathrm{t}\dot{\boldsymbol{\eta}}^n) - (1-\zeta)\mathbf{k}(\boldsymbol{\eta}^n)\,, \tag{4.4}$$

and the right-hand side of the velocity and acceleration:

$$\begin{aligned} \mathsf{rhsW} &= \dot{\boldsymbol{\eta}}^n + \delta \mathrm{t}(1-\theta)\ddot{\boldsymbol{\eta}}^n\,, \\ \mathsf{rhsA} &= \frac{2}{\zeta \delta \mathrm{t}^2}(\boldsymbol{\eta}^n + \delta \mathrm{t}\dot{\boldsymbol{\eta}}^n) + \frac{(1-\zeta)}{\zeta}\ddot{\boldsymbol{\eta}}^n\,. \end{aligned} \tag{4.5}$$

iterate:

Method `iterate` calls template NLR where a Newton method used to linearize the problem is implemented . The main lines of code of the method are as follows:

```
M_BCh = bch;
status = nonLinRichardson( *M_disp, *this, abstol, reltol, maxiter, etamax,
                           linesearch, M_out_res, M_data->dataTime()->getTime() );
updateVelocityandAcceleration();
```

where, in the first line, the boundary conditions of the problem specified on main file structure are loaded, while in the second line the NLR template is called. NLR return 0 when Newton converges and 1 when Newton failed to converge. Moreover in NLR two method of NLSS are called : evalResidual and solveJac that are described in the following subsection.

NLR template:

NLR template implements a Newton method for the linearization of the nonlinear structural problem. As mentioned, it is called by NLSS's method, iterate. NLR, recalls two methods of NLSS: evalResidual and solveJac. In particular, NLR, initially recalls evalResidual, to compute the residual, and tolerances:

```
functional.evalResidual( residual, sol, iter );
Real normRes      = residual.NormInf();
Real stop_tol     = abstol + reltol*normRes;
```

Then, enters into the following while loop:

```
while ( normRes > stop_tol && iter < maxit ){
...
functional.solveJac(step, -1.*residual, linearRelTol);
sol += step;
functional.evalResidual( residual, sol, iter);
...}
```

where solveJac(step, -1.*residual, linearRelTol) is called. It computes the new Jacobian matrix and solves the linearized system $\mathrm{J}_{\mathcal{L}}(\boldsymbol{\eta}^{(k)})\boldsymbol{\delta\eta}^{(k)} = -\mathcal{L}(\boldsymbol{\eta}^{(k)})$, then updates the solution $\boldsymbol{\eta}^{(k+1)} = \boldsymbol{\eta}^{(k)} + \boldsymbol{\delta\eta}^{(k)}$ and recalls NLSS's method evalResidual.

evalResidual:

Method `evalResidual` computes the residual $\mathcal{L}(\boldsymbol{\eta}^{(k)})$ of algebraic system: $\mathrm{J}_{\mathcal{L}}(\boldsymbol{\eta}^{(k)})\boldsymbol{\delta\eta}^{(k)} = -\mathcal{L}(\boldsymbol{\eta}^{(k)})$. In particular, the NLSS's method computeMatrix is called. It updates the nonlinear stiffness term $\mathbf{k}(\boldsymbol{\eta})$ which depends on the solution:

```
computeMatrix(this->M_matrix_stiff, this->M_vector_stiff, sol, 1.);
```

Then, it updates the boundary conditions:

```
if ( !this->M_BCh->bdUpdateDone() )
this->M_BCh->bdUpdate( *this->M_FESpace->mesh(), this->M_FESpace->feBd(),
                       this->M_FESpace->dof() );
```

and applies these to the residual in the following form:

```
*M_matrix_stiff += *M_mass;
bcManageMatrix( *M_matrix_stiff, *M_FESpace->mesh(), M_FESpace->dof(),
                *M_BCh, M_FESpace->feBd(), 1.0 );
vector_type rhsFull(*M_rhsNoBC, Unique);
bcManageVector( rhsFull, *M_FESpace->mesh(), M_FESpace->dof(),
                *M_BCh, M_FESpace->feBd(), M_data->getTime(), 1.0 );
```

ie. on the matrix of the system and on the right-hand side separately. Finally, the residual with boundary conditions is computed:

```
res  = *M_matrix_stiff*sol;
res -= *M_rhs;
```

solveJac:

SolveJac method, serves to solve the linearized system. First, it calls NLSS's method updateJacobian:

```
updateJacobian( *this->M_disp, this->M_jacobian );
```

to update the Jacobian matrix. Then, the boundary conditions are applied on Jacobian matrix, calling NLSS's method applyBoundaryConditions:

```
applyBoundaryConditions( *this->M_jacobian, rhsFull, BCh);
```

Finally the Jacobian matrix is defined as iteration matrix and the algebraic system is solved, calling a linear solver AztecOO and the preconditioners package Ifpack.

```
this->M_linearSolver->setMatrix(*this->M_jacobian);
int numIter = this->M_linearSolver->solveSystem( rhsFull, step, this->M_jacobian );
```

In particular the linear system is solved using GMRES method and additive Schwarz method as preconditioner.

4.2.2 Structural problem: Solution and code performances

In this brief subparagraph we explain the steps for the solution of the structural problem from an implementative point of view. In particular, the Newton loop and the iterative solution of the linearized algebraic system are shown. Special attention is given to computational costs of the single step to indicate which is the critical points and where it is possible to improve the solver. We have evaluated mean quantities to perform a standard test such as the simulation presented in chapter two: the inflation of the hollow cylinder. In particular the time-steps performed are 25 for each simulation and the internal pressure is a physical shape.

The mesh used is more sparse than the mesh used in Chapter 2 for the hollow cylinder inflation test. It is composed by 1360 nodes and the finite-element are P1. This choice allows to play the test quickly and take the values of CPU-time averages. In table 4.1 it is possible to see the mesh properties in details.

Table 4.1: Mesh properties of *vessel20*

Nodes	**Triangles**	**Thetrahedra**	**Length**	**Inner radius**	**Outer radius**
1360	1760	4800	5 cm	0.5 cm	0.6 cm

We are interested in particular to the following quantities:

- Computational cost to build the system;

- Computational cost to update the Jacobian;
- Number of Newton iteration per time-step;
- Newton residual vs. Newton iteration;
- Dimensionless CPU time to perform a test using three different structural models.

Finally, we have evaluated the correct parallelization of the code by scalability test on *Lagrange* cluster.

Newton loop

As previously mentioned, NLSS perfom the following sub-routines to complete a single time-step.

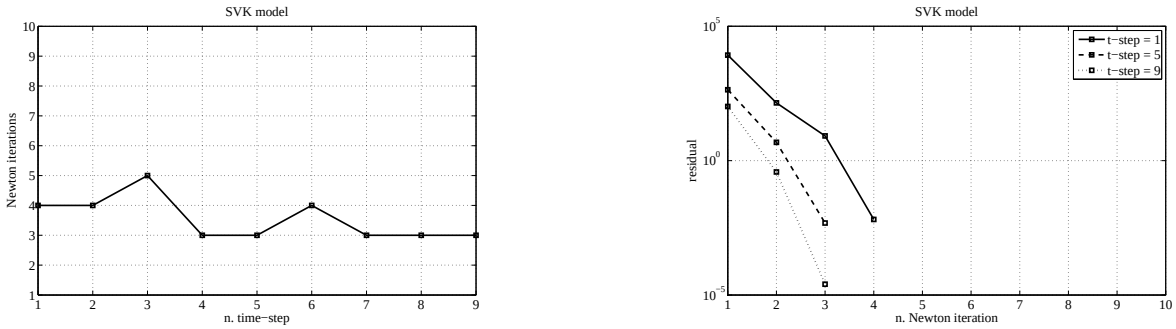
1. Initialization of displacement, velocity and acceleration (first time-step only) or updating velocity and acceleration;
2. Building the constant matrices of mass and linear stiffness (first step only);
3. Updating the right-hand side;
4. Enter on Newton Loop until to convergence:
   (a) Computing residual and compare it with the tolerance. If residual is bigger then tolerance:
   (b) Updating Jacobian;
   (c) Solving linearized system using GMRES (see next subparagraph) till to convergence;
5. When convergence of Newton's loop is done, go to the next time-step.

A simple table 4.2 is reported below, which specifies the computational costs for each material to perform the steps above.

Steps	**St.Venant-Kirchhoff**	**Neo-Hookean**	**Exponential**
1-2-3	0.5731 (2.6212 s)	0.0472 (1.1600 s)	0.0362 (1.1525 s)
4-(a)	0.4220 (1.9300 s)	0.0215 (0.5275 s)	0.0170 (0.5400 s)
4-(b)	1.0000 (4.5738 s)	1.0000 (24.5837 s)	1.0000 (31.8463 s)
4-(c)	0.0626 (0.2863 s)	0.0155 (0.3800 s)	0.0085 (0.2717 s)

Table 4.2: Computational costs to perform the sub-routines necessary to perform a single time-step of the structural problem

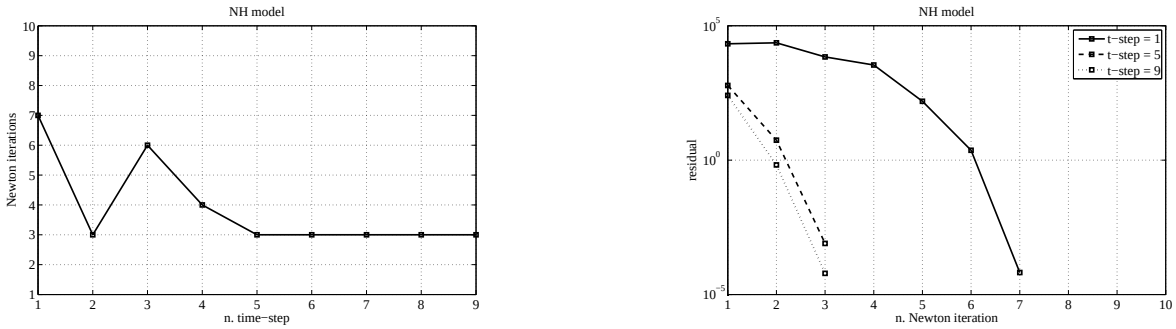
Moreover, in figure 4.2(a)-4.2(b), we show the trend of the residual versus the number of Newton iterations and the Newton iterations versus time-step for St.Venant-Kirchhoff structural model. In the figures 4.3 and 4.4 we show the same quantities for the other two model, respectively Neo-Hookean and Exponential. It is possible to observe that the tolerance of Newton loop is, approximately, 10^{-2} and the St.Venant-Kirchhoff model shows the best performance in terms of iteration per time-step.



(a) Number of iteration per time-step

(b) Residual vs. Newton iteration

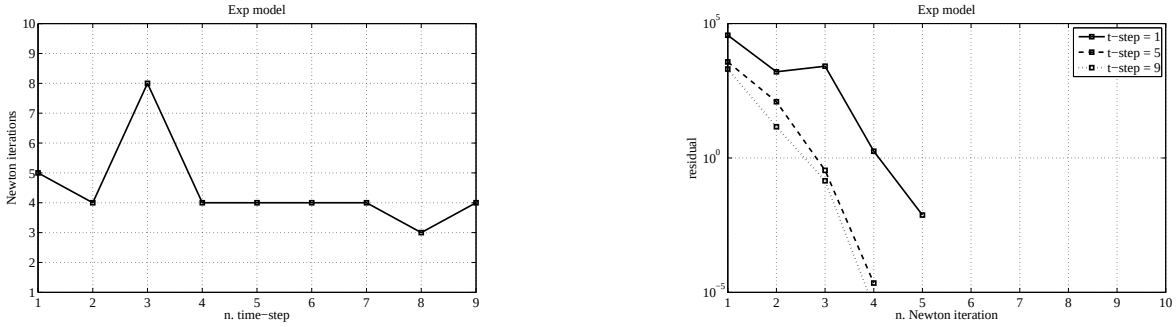
Figure 4.2: St.Venant-Kirchhoff model: Trend of the residual vs. Newton iteration and number of iteration per time-step



(a) Number of iteration per time-step

(b) Residual vs. Newton iteration

Figure 4.3: Neo-Hookean model: Trend of the residual vs. Newton iteration and number of iteration per time-step



(a) Number of iteration per time-step

(b) Residual vs. Newton iteration

Figure 4.4: Exponential model: Trend of the residual vs. Newton iteration and number of iteration per time-step

Plus the cost of the individual methods, also the convergence properties affect the computational costs required to the CPU. Combining the two things, it is interesting to compare the dimensionless CPU time required for the simulations performed with three structural models 4.3. The highest CPU time, is required from Exponential model. In fact, this material presents the highest number of Newton iterations per time-step. Moreover, the major contribute to the CPU time required is the computation of the Jacobian matrix that is highest for the exponential model.

Table 4.3: Dimensionless CPU-time comparison for a standard structural simulation using three different structural models

Model	Total CPU-time	Jacobian CPU-time	Total Newton iterations
Exp	1.00	1.00	40
NH	0.68	0.73	35
SVK	0.19	0.20	32

Solution of algebraic system

A very important part for the solution of a finite-element problem is represented by the solution of algebraic system. In fact, as anticipated, for large algebraic systems it is possible to have an ill-conditioned iteration matrix. Hence, it is necessary to use adequate tools for resolving the problem. Moreover, if we use parallel-computing, preconditioning and solution of algebraic system must be made with great care.

In the nonlinear structural problem, for each iteration of the Newton method the Jacobian matrix of the system and the residual are re-calculated, and the following problem is solved:

$$\mathrm{J}_{\mathcal{Z}}(\boldsymbol{\eta}^{(k)})\boldsymbol{\delta\eta}^{(k)} = -\mathcal{Z}(\boldsymbol{\eta}^{(k)}) \tag{4.6}$$

In general case, $\mathrm{J}_{\mathcal{Z}}(\boldsymbol{\eta}^{(k)})$ is a large sparse matrix $n \times n$ and the residual $\mathcal{Z}(\boldsymbol{\eta}^{(k)})$ is a column vector of size n. The solution of this algebraic system is performed by an iterative Krylov method and in

particular by GMRES, using as precontitioning the additive Schwarz method. We now analyze the two steps that we have to perform the solution of (4.6) using simplified notation $\mathrm{J} = \mathrm{J}_{\mathcal{Z}}(\boldsymbol{\eta}^{(k)})$ and $\mathbf{r} = -\mathcal{Z}(\boldsymbol{\eta}^{(k)})$.

- **Additive Schwarz preconditioning**: Is a domain decomposition method DD. In this type of preconditioners the computational domain Ω_{h} is divided into m smaller subdomains such that the original problem is reformulated within each subdomain Ω_i. To guarentee the correct solution of the original problem additional interface conditions are added. In particular the subproblems is coupled one to another by the values of the unknown solution at subdomain interface. Hence to solve the preconditioning global problem it is necesessary to solve the single preconditioning problem for each subdomain respecting to the coupling conditions. The additive-Schwarz preconditioner reads in the following form:

$$\mathrm{P}_{\mathrm{AS}}^{-1} = \sum_{i=1}^{m} \mathrm{P}_i \, \mathrm{J}_i^{-1} \, \mathrm{R}_i \,, \tag{4.7}$$

  where J_i is a partition of J. In particular the number of subdomains m corresponds to the CPUs involved in the computation. DD additive Schwarz preconditioners are used in particular when the algebraic system is solved with iterative Krylov type methods. GMRES method is part of this category. For the last test case on the hollow cylinder, the mean time to compute the preconditioner using 1 and 2 CPUs is evaluated, for each material. The results are reported in the table 4.4(a).

- **GMRES method**: GMRES (Generalized Minimal RESiduals) is a method based on Krylov subspaces' iterations. It is used for the solution of algebraic system (4.6). In particular, if $\mathrm{J} \in \mathbb{R}^{n \times n}$ and $\mathbf{r} \in \mathbb{R}^n$ a Krylov space of index r, is defined as the following set:

$$\mathcal{K}_r = \mathrm{span}(\mathbf{r}, \mathrm{J}\mathbf{r}, ..., \mathrm{J}^{r-1}\mathbf{r}) \subseteq \mathbb{R}^n \tag{4.8}$$

  The projection method on Krylov's subspaces gives for each r an approximated solution $\delta\boldsymbol{\eta}^{\ell}$ of the linear system (4.6) belonging to $\mathcal{K}_r$. If all vectors $\mathrm{J}^k\mathbf{r}$ for k=1,2,...,n-1 are indipendent, the Krylov space corresponds to $\mathbb{R}^n$ and the algorithm terminates in n steps. While, if the above condition is not verified, $\mathcal{K}_r < \mathbb{R}^n$ and the algorithm converges in a number of iterations less than n. The criterion to choice the vector $\delta\boldsymbol{\eta}^{\ell}$, distinguishes the GMRES method from the other methods. In particular, at each iteration ℓ, a vector $\delta\boldsymbol{\eta}^{\ell} \in \mathcal{K}_{\ell}$ is chosen. In particular, it minimizes the Euclidean norm of the residual defined as $\mathbf{r}^{\ell}_{\mathrm{gmres}} = \mathrm{J}^{\ell}\delta\boldsymbol{\eta}^{\ell} + \mathbf{r}^{\ell}$.

As described above, the solution of the linearized algebraic system does not require an high computational cost relative because the choice of preconditioner and linear solver is efficient. In fact, it is possible to view in table 4.4 that the preconditioning of system is scalable (4.4(a)) and gives a good condition number (4.4(b)).

Table 4.4: Comparison of CPU-time using 1 and 2 CPUs for preconditioniing and solving the linearized system using additive Schwarz as preconditioner and GMRES as linear solver and evaluation of condition number

(a) CPU-time to compute the preconditioner

	1 CPU	2 CPU
CPU-time SVK	0.24 s	0.16 s
CPU-time NH	0.24 s	0.17 s
CPU-time EXP	0.23 s	0.16 s

(b) Estimate condition number for additive-Schwarz preconditioner

	1 CPU	2 CPU
Cond-n. SVK	1.06495	1.06544
Cond-n. NH	1.07207	1.07420
Cond-n. EXP	1.08335	1.08519

(c) CPU-time to compute the preconditioner and solve the linearized system

	1 CPU	2 CPU
CPU-time SVK	0.26 s	0.41 s
CPU-time NH	0.25 s	0.43 s
CPU-time EXP	0.27 s	0.40 s

Furthermore, the deterioration of spectral properties of the jacobian matrix using more CPUs is low as shows in table 4.4(c). Table 4.5 shows the advantages of using *ad hoc* preconditioner instead of reusing preconditioner in terms of CPU-time (4.5(a)-4.5(b)) and in terms of GMRES iterations (4.5(c)-4.5(d)).

Table 4.5: CPU-time to solve the linearized system using and reusing preconditioner and evaluation of GMRES iteration using 1 and 2 CPUs

(a) CPU-time to solve the linearized system using specific preconditioner

	1 CPU	2 CPU
CPU-time SVK	0.03 s	0.25 s
CPU-time NH	0.02 s	0.25 s
CPU-time EXP	0.03 s	0.25 s

(b) CPU-time to solve the linearized system reusing preconditioner

	1 CPU	2 CPU
CPU-time SVK	0.26 s	0.43 s
CPU-time NH	0.71 s	0.92 s
CPU-time EXP	1.25 s	1.36 s

(c) Number of GMRES iterations using specific preconditioner

	1 CPU	2 CPU
GMRES-iter SVK	1	15
GMRES-iter NH	1	18
GMRES-iter EXP	1	18

(d) Number of GMRES iterations reusing preconditioner

	1 CPU	2 CPU
GMRES-iter SVK	18	26
GMRES-iter NH	49	65
GMRES-iter EXP	82	99

Scalability on multiple processors

It is very important to see the correct parallelization of the code because one of the aim of this work is to have a code avalaible in a parallel version. To carry out the evaluation we used two indices:

1. **Speed-up**:

$$S_p = \frac{T_1}{T_p} \tag{4.9}$$

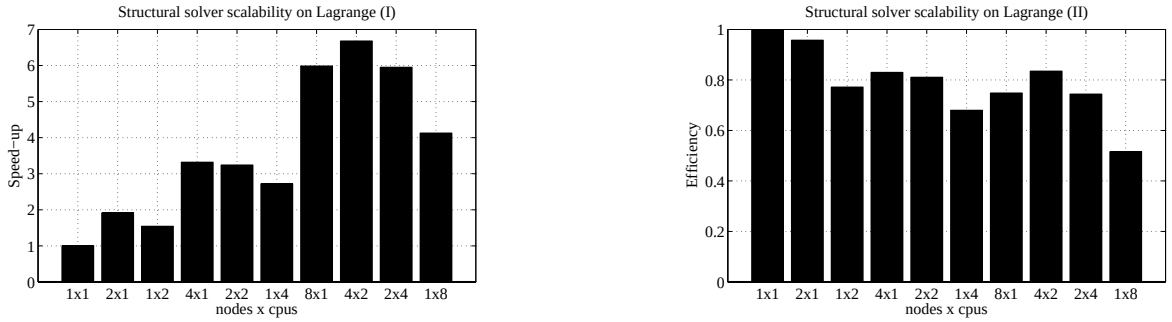
   where p is the number of processors, T_1 is the computing time using only one processor and T_p is the computing time using p processors.

2. **Efficiency**:

$$E_p = \frac{S_p}{p} \tag{4.10}$$

   that characterizes the parallel performance.

In figure 4.5(a) we show the Speed-up index, while in figure 4.5(b) we show the second index, Efficiency. The scalability test was done on Lagrange cluster of Cilea. It is composed of 208 dual-processor nodes and each processor is an Intel Xeon quad-core X5460 with 3.166 GHz. In table 4.6 we show the principal characteristic of the cluster.



(a) Speed-up index on Lagrange

(b) Efficiency index on Lagrange

Figure 4.5: Scalability test of the structural solver on Lagrange cluster

Table 4.6: Technical characteristics of Lagrange cluster

N.nodes	CPUs	Processor	Clock	Ram	Capacity	Interconnection
208	1664	2 Intel Xeon X5400	3.166 GHz quad-core	16 GB	13 TB	1 Infiband 4X DDR 2 Gigabit Ethernet

for more information about software LifeV on Lagrange cluster, see [57].

4.3 FSI solver implementation and code performances

In this section we present the main useful methods for the solution of the fluid-structure problem. Unlike the structural solver, we should start from the main file, following the branching of the calls in the various templates that form the FSI solver. After an overview of the methods needed to carry out a time step, we must analyze the computation time needed to perform the main processes of the EJ algorithm. It also makes a comparison between the computation time to perform a simulation with linear and non-linear structure.

4.3.1 Methods description

In the following list, the main methods to solve a fluid-structure problem are shown. In particular we enter into implementation details, showing the most important rows of code and their tasks.

FSISolver

Class FSISolver contains the main methods are recalled into the *main* file that defines a FSI problem. In particular the most important methods are the following:

-setup:

Method setup serves to initialize the solvers of each sub-problem and to make the preliminaries routines such as parameters inizialization. It recalls three methods of *FSIOperator* with three code lines below:

```
M_oper->setupFluidSolid();
M_oper->setupSystem();
M_oper->buildSystem();
```

-iterate:

It is the main method of *FSISolver* and it is used to start the external loop of FSI problem. In particular it recalls the template *nonLinRichardson* with the following code lines:

```
status = nonLinRichardson( *lambda,
                           *M_oper,
                           M_data->absoluteTolerance(),
                           M_data->relativeTolerance(),
                           maxiter,
                           ...
                           M_data->dataFluid()->dataTime()->getTime() );
```

where absolute and relative tolerance and other quantities are defined into *data* file.

FSIOperator

FSIOperator contains some methods to manage the FSI problem such as the coupling conditions or the solvers inizialization. In particular as previously mentioned, *FSISolver* recalls three method of *FSIOperator* that are reported below.

-setupFluidSolid:

It resets all subproblem solvers and calls their setup methods.

```
M_meshMotion.reset   ( new meshmotion_raw_type( *M_mmFESpace, M_epetraComm ) );
M_fluid.reset        ( new fluid_raw_type( M_data->dataFluid(), *M_uFESpace,
                          *M_pFESpace, M_epetraComm, numLM ) );
...
M_solid.reset( solid_raw_type::StructureSolverFactory::
               instance().createObject( M_data->dataSolid()->solidType( ) ) );
M_solid->setup( M_data->dataSolid(), M_dFESpace, M_epetraComm );
```

In particular, here the solid solver is defined as a factory, where VenantKirchhofSolver is the master and *LinearVenantKirchhofSolver* and *NonLinearStructureSolver* are the derived classes.

-setupSystem:

It interfaces the data files of each subproblem with the FSI solver, using the following lines of code:

```
M_fluid->setUp( M_dataFile );
M_meshMotion->setUp( M_dataFile );
...
M_solid->setDataFromGetPot( M_dataFile );
```

-buildSystem:

It recalls the buildSystem methods of fluid and solid subproblem.

```
M_fluid->buildSystem();
M_solid->buildSystem();
```

BuildSystem method for solid subproblem is the same described in the paragraph 4.2.

-updateSystem:

It recalls the updateSystem methods of each subproblem with the following code lines:

```
M_meshMotion->updateSystem();
...
transferMeshMotionOnFluid(M_meshMotion->disp(), *this->M_dispFluidMeshOld);
```

```
if( M_fluid->solution().get() )
    M_un.reset( new vector_type( *M_fluid->solution() ) );
...
this->M_solid->updateSystem();
```

Moreover, it transfers the mesh displacement on fluid subproblem.

-couplingVariableExtrap:

It manages the interface displacement and velocity that have to be passed at the next time-step:

```
*M_lambda      = lambdaSolid();
*M_lambdaDot   = lambdaDotSolid();
...
```

exactJacobianBase

As mentioned in chapter 3, *exactJacobianBase*, has some important methods. Here we show two methods that represents the subroutines to perform a time-step. First method (evalResidual) is called from the template *nonLinRichardson*, while the second method (eval) is called by evalResidual. In particular eval represents the external loop of the FSI problem. In the following bulleted list we show briefly the most important rows of these methods.

-evalResidual:

This method computes the residual of FSI problem needs to perform the solution of linearized problem and also used as stopping criterion. In particular the main rows are the following:

```
this->setLambdaSolidOld(disp);
eval(disp, iter);
res  = this->lambdaSolid();
res -=  disp;
```

where in the second line eval method is recalled and in the last two lines residual is defined.

-eval:

This method represents the external loop. Here solid and fluid subproblems is recalled and the coupling conditions are treated.

```
this->M_meshMotion->iterate(*M_BCh_mesh);
this->transferMeshMotionOnFluid(M_meshMotion->disp(), this->veloFluidMesh());
this->M_fluid->iterate( *M_BCh_u );
this->transferFluidOnInterface(-this->M_fluid->residual(), sigmaFluidUnique);
this->M_solid->iterate( M_BCh_d );
this->transferSolidOnInterface(this->M_solid->disp(),     lambdaSolidUnique);
this->transferSolidOnInterface(this->M_solid->vel(),      lambdaDotSolidUnique);
this->transferSolidOnInterface(this->M_solid->residual(), sigmaSolidUnique);
```

4.3.2 FSI problem: Solution and code performances

As for the structural problem, here we explain the steps for the solution of the FSI problem from an implementative point of view. In particular, the solution of the harmonic, fluid and solid subproblems are shown. Special attention is given to computational costs of the single step to indicate which is the critical points and where it is possible to improve the solver. We have evaluated mean quantities to perform a standard test with a physical sine-wave inlet pressure. The meshes used are *vessel20* for the structure and *tube20* for the fluid previously described in the table 3.3.
Principal quantities investigated are reported in the bulleted list below:

- External Newton loop iterations per time-step;
- External Newton loop residual vs. external Newton loop iteration;
- Nonlinear structure Newton loop iterations per time-step;
- Nonlinear structure Newton loop residual vs. nonlinear structure Newton loop iteration;
- Dimensionless CPU time to perform a test using three different structural models and comparison with linear structure;
- Dimensionless CPU time of the GMRES matrix-free to solve the interface problem for three different structural models and comparison with linear structure.

External and nonlinear structure Newton loop

EJ algorithm has two principal loops. The first one is a Newton loop on nonlinear interface problem, called external or outer loop. While the second one is a Newton loop on nonlinear structure problem (the same presented in chapter two) and it is called internal or inner loop. In particular, the algorithm used perform the following sub-routines per time-step.

1. Computing constant matrices for harmonic extension, fluid and solid subproblems and initializing parameters;
2. Updating system and right-hand side;
3. Loading adsorbing boundary conditions and initial data;
4. Entering in the **external Newton loop** until to convergence:
   (a) Harmonic problem:
      i. Applying boundary conditions and solving linear system;
   (b) Fluid problem:
      i. Moving mesh;[63]
      ii. Updating mass term on right-hand side;
      iii. Computing constant matrices and finalizing them;
      iv. Updating and applying boundary conditions;
      v. Solving linear system;

(c) Solid problem:

i. Building the constant matrices of mass and linear stiffness;

ii. Updating the right-hand side;

iii. Enter on **internal Newton loop** until to convergence:

A. Computing residual and compare it with the tolerance. If residual is bigger then tolerance:

B. Updating Jacobian;

C. Solving linearized system using GMRES until to convergence;

(d) Evaluating of external Newton loop residual and comparing it with the tolerance. If the residual is smaller than the tolerance, go to the next time-step, however, if the residual is bigger than the tolerance:

(e) Solving the linearized interface problem with matrix free GMRES method and return to step (4).

As for the structural solver (NLSS) it is interesting to evaluate the CPU time required to complete the single processes because to improve the solver and evaluate its performances[2]. In table 4.7 the computational costs for each material are reported.

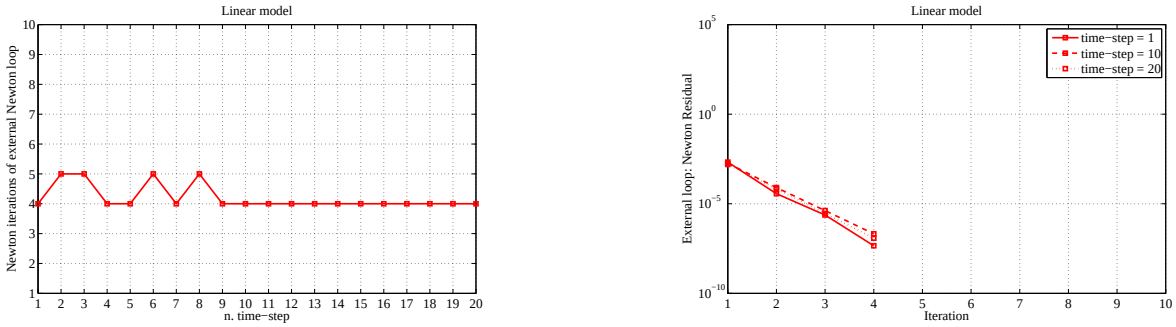
Table 4.7: Computational costs to perform the sub-routines necessary to perform a single time-step of the FSI problem

Steps	**Linear**	**St.Venant-Kirchhoff**	**Neo-Hookean**	**Exponential**
1-2-3	1.0000 (4.4121 s)	0.4424 (6.4200 s)	0.1142 (4.8701 s)	0.0937 (4.9307 s)
4-(a)-i	0.0116 (0.0513 s)	0.0035 (0.0510 s)	0.0012 (0.0511 s)	0.0010 (0.0510 s)
4-(b)-i–iv	0.8416 (3.7134 s)	0.2536 (3.6801 s)	0.0868 (3.6981 s)	0.0705 (3.7098 s)
4-(b)-v	0.1955 (0.8625 s)	0.0580 (0.8415 s)	0.0200 (0.8532 s)	0.0160 (0.8430 s)
4-(c)	0.6303 (0.2781 s)	1.0000 (14.5102 s)	1.0000 (42.6312 s)	1.0000 (52.5907 s)
4-(d)–(e)	0.9250 (4.0812 s)	0.5411 (7.8520 s)	0.1834 (7.8191 s)	0.1521 (8.0001 s)

Figures 4.6(a), 4.7(a), 4.8(a) and 4.9(a) represent the number of Newton iterations per time-step of the external and internal loops for three nonlinear structural models and for linear elasticity. It

[2]Analysis of the solid problem has already been carried out in section 4.2. Then, for it, the total CPU-time is evaluated only

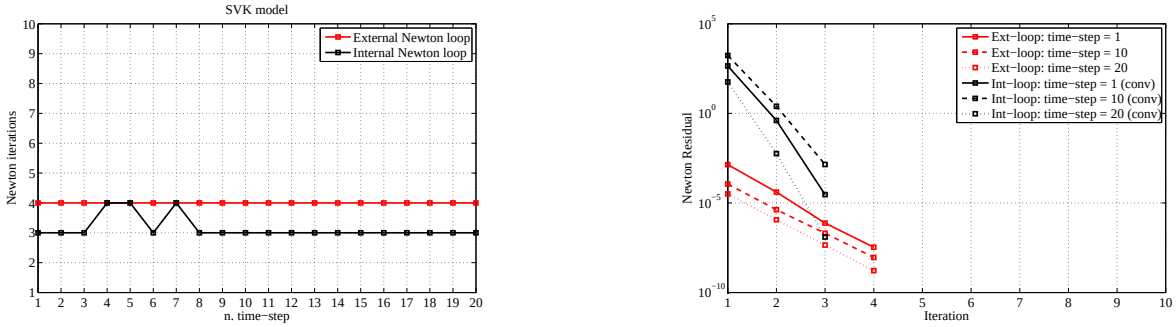
is interesting to see that the number of outer Newton iterations for Neo-Hookean and Exponential models are the same of linear elasticity and St.Venant-Kirchhoff model.



(a) Number of iteration per time-step

(b) Residual vs. Newton iteration

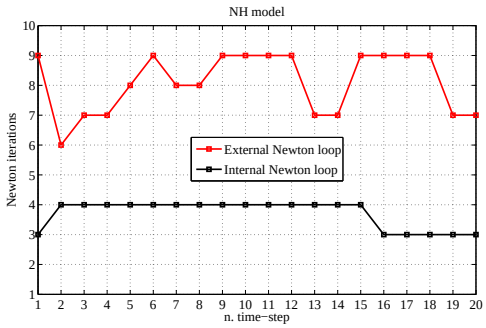
Figure 4.6: Linear elasticity: Trend of the residual vs. Newton iteration and number of iteration per time-step



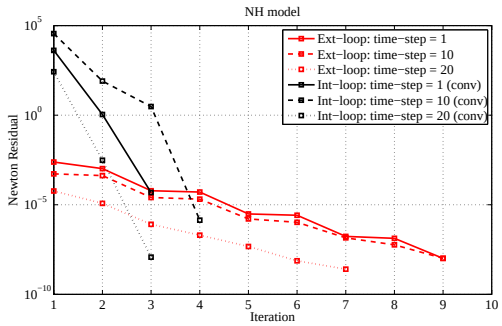
(a) Number of iteration per time-step

(b) Residual vs. Newton iteration

Figure 4.7: St.Venant-Kirchhoff model: Trend of the residual vs. Newton iteration and number of iteration per time-step

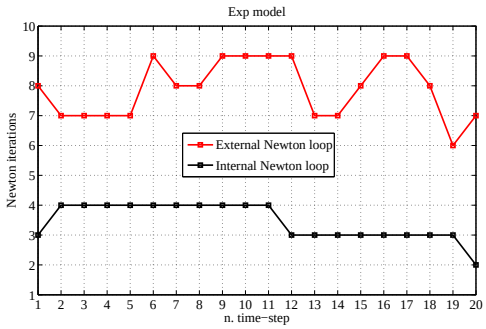


(a) Number of iteration per time-step

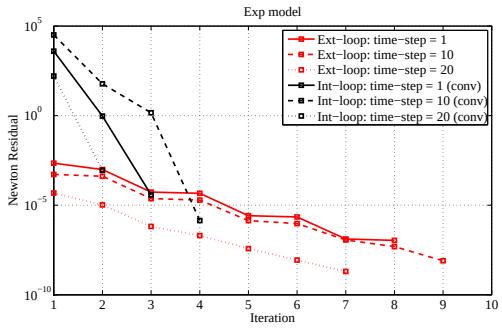


(b) Residual vs. Newton iteration

Figure 4.8: Neo-Hookean model: Trend of the residual vs. Newton iteration and number of iteration per time-step



(a) Number of iteration per time-step



(b) Residual vs. Newton iteration

Figure 4.9: Exponential model: Trend of the residual vs. Newton iteration and number of iteration per time-step

However, the number of inner Newton iterations for Neo-Hookean and Exponential models are bigger than other two structural models. This finding is consistent with the structural results obtained in Chapter 2 where we see an increase in the number of Newton iterations per time-step for the two models NH and Exp.

The convergence properties of the external loop are comparable with the results obtained in [19]. Clearly, the computational cost to make a simulation with nonlinear structural models is greater than the use of linear elasticity to describe the structural model. In particular in the table 4.8 it is possible to compare the CPU-time to perform a standard simulation with 20 time-steps and with the space discretization described previously.

Here, it in not possible to perform a scalability test on the overall problem, because the interface map is still not parallel. In fact, the fluid sub-problem only is parallel at the moment.

Table 4.8: Dimensionless CPU-time comparison for a standard FSI simulation using 4 different structural models

Model	Total CPU-Time	Total FSI Newton iterations
Exp	1 (16551 s)	158
NH	0.8563 (14181 s)	162
SVK	0.1730 (2864 s)	80
Linear	0.0849 (1406 s)	84

Chapter 5

FSI: haemodynamic results

One of the most important hemodynamic parameters is certainly the wall-shear stress (WSS). In this chapter we want to evaluate this parameter to the healthy carotid arteries using linear elasticity and 3 nonlinear structural models to compare the results. Moreover, we compare also the primitive variables such as fluid velocity and pressure and structural displacement.

5.1 WSS parameter

The role played by the parameter WSS in hemodynamics is well known in the literature [58], [59]. Recent studies have classified it as a parameter of significant interest in the atherosclerotic disease [60], [61]. In particular, with regard to vessel bifurcations, such as the carotid arteries shown in this chapter, we know that atherosclerosis affects in particular regions of the external wall immediately after the bifurcation [62], [63]. In these areas, WSS acting on the endothelial cell surface [60], is weaker than other regions of the vessel. Two areas of WSS values have been identified, that appear to induce opposite effects on the arterial wall:

- WSS > 15 dyne/cm^2: Induces endothelial quiescence and an atheroprotective gene expression profile.
- WSS < 4 dyne/cm^2: It is prevalent at atherosclerosis-prone sites, stimulates an atherogenic phenotype.

It is also possible to demonstrate the correlation between arteriovenous malformations in cerebral arteries and WSS [64]. Hence, a proper evaluation of this hemodynamic parameter can then provide relevant clinical data for a correct treatment of artheriosclerosis and correlated patologies in arteries. The WSS index represents the surface stress on the wall induced by fluid dynamic field:

$$\begin{aligned} \widetilde{\mathrm{WSS}} &= -\mu[(\boldsymbol{\nabla}\mathrm{u}_x \cdot \mathbf{n}_\Sigma)\mathbf{e}_\mathrm{x} + (\boldsymbol{\nabla}\mathrm{u}_y \cdot \mathbf{n}_\Sigma)\mathbf{e}_\mathrm{y} + (\boldsymbol{\nabla}\mathrm{u}_z \cdot \mathbf{n}_\Sigma)\mathbf{e}_\mathrm{z}]\,, \\ \mathrm{WSS} &= \widetilde{\mathrm{WSS}} - (\widetilde{\mathrm{WSS}} \cdot \mathbf{n}_\Sigma)\mathbf{n}_\Sigma\,, \end{aligned} \tag{5.1}$$

where u_x, u_y, u_z are the velocity components in cartesian coordinates, $\mathbf{n}_\Sigma$ is the normal vector with respect to the interface and $\mathbf{e}_\mathrm{x}$, $\mathbf{e}_\mathrm{y}$, $\mathbf{e}_\mathrm{z}$ are the cartesian versors.

In the figure 5.1, taken from [60], it is possible to see the mechanism of plaque formation. While, in the table 5.1, it is possible to show the range of the WSS parameters for healty arteries and pathological arteries.

Table 5.1: Range of the WSS parameter for different pathological and non-pathological cases

	Normal artery	**Atherosclerosis**	**Thrombosis** (complex plaque)
Range WSS[dyne/cm^2]	10÷70	-4÷4	70÷ >100

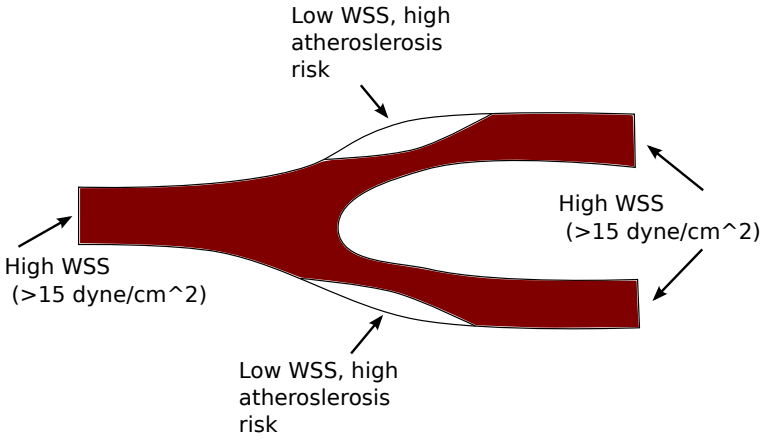


Figure 5.1: Typical values of WSS for carotids

5.2 FSI simulation results on carotid

From the discussion in Section 5.1, the first parameter of interest is the WSS. However, it is interesting to evaluate the values of velocity and pressure of the fluid, and the displacement of structure on certain sections of the carotid. In particular, the differences between linear elasticity and nonlinear structural models are emphasized. We have evaluated the structure displacement, the mean fluid pressure and the mean fluid velocity at three different sections along the axis of the carotid, as shown in figures 5.2-5.3-5.4.

We have used a space discretization for fluid (*carotid_fluid*) and structure (*carotid_solid*) with the characteristics reported in the table 5.2.

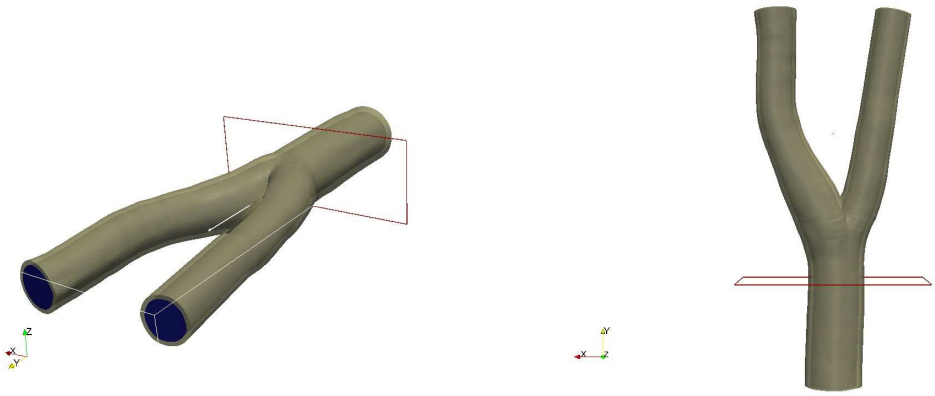
	Nodes	**Triangles**	**Thetrahedra**	**Length**[cm]
carotid_solid	24100	43388	78640	5.42
carotid_fluid	20072	17704	92188	5.42

We have used P1-P1-P1 finite-element to discretize structure displacement, fluid pressure and fluid velocity. In particular, a stabilization term is used to solve correctly the fluid sub-problem. Moreover, we have used $t_0 = 0s$, $t_N = 0.01s$ and $\delta t = 0.0005s$.

The boundary conditions applied to the problem are the following:

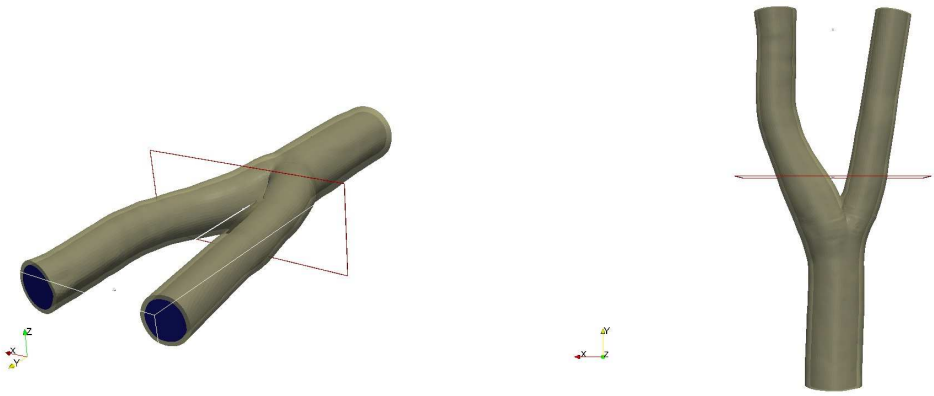
- Structure basis: embedded;
- Structure outer surface: Stress free;
- Structure inner surface: Neumann condition from the fluid subproblem;
- Fluid inlet: $p = 1.332e4\sin(\frac{\pi t}{0.003})$ until $t < 0.003s$. $p = 0$ for $t > 0.003s$;
- Fluid outlet: Homogeneous Neumann condition;
- Fluid inner surface: Dirichlet condition from the structure subproblem.

Remark 6. Using higher pressures than $p = 1.332e4\sin(\frac{\pi t}{0.003})$, we have found some instabilities in inflow section, still under investigation.



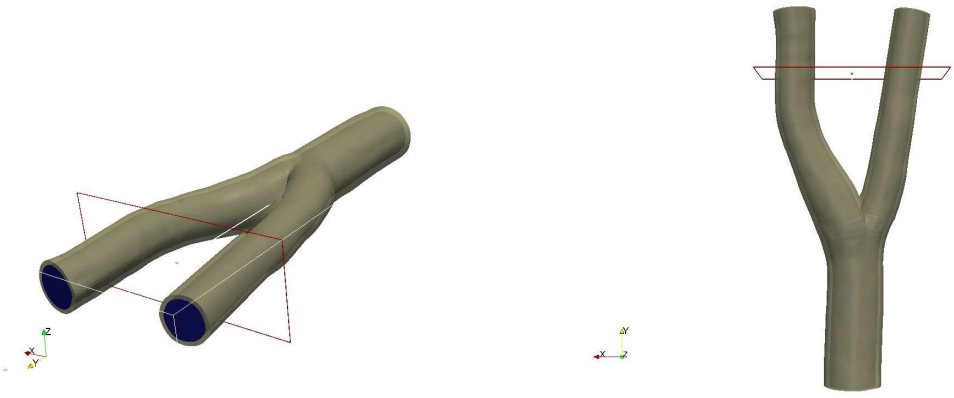
(a) Isometric visual of the first section (b) Plane visual of the first section

Figure 5.2: First section of the carotid



(a) Isometric visual of the first section (b) Plane visual of the first section

Figure 5.3: Second section of the carotid



(a) Isometric visual of the first section (b) Plane visual of the first section

Figure 5.4: Third section of the carotid

Figure 5.5 is the maximum value of the WSS parameter for each time-step. It is possible to note how the WSS has the minimum in proximity of the inlet of the bifurcation. In fact, the wave pressure arrives at the bifurcation about in 0.004-0.0045 seconds. This results is in accord with the theory [60].

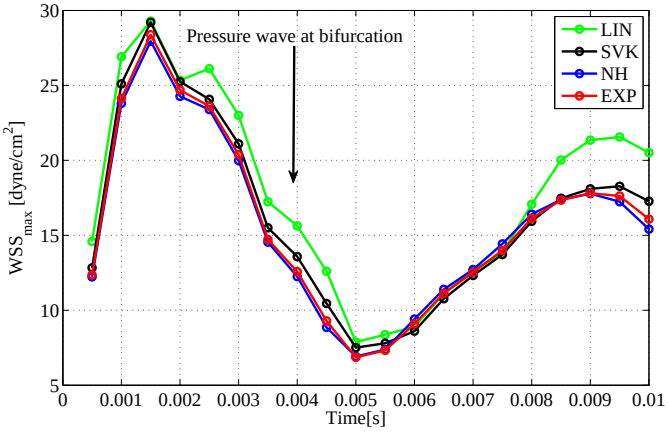
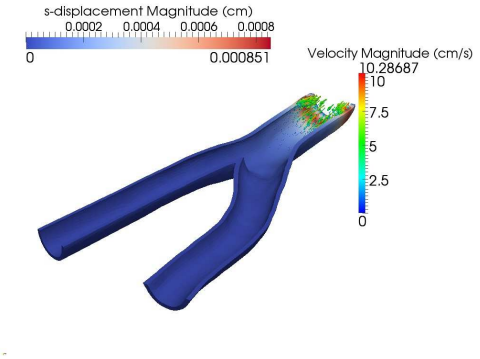
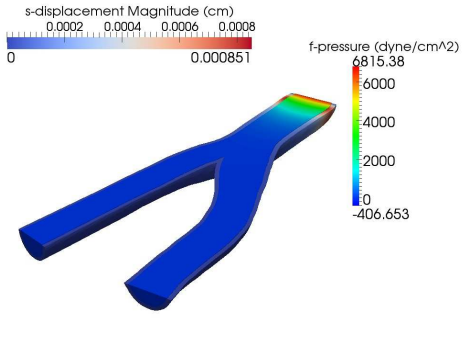


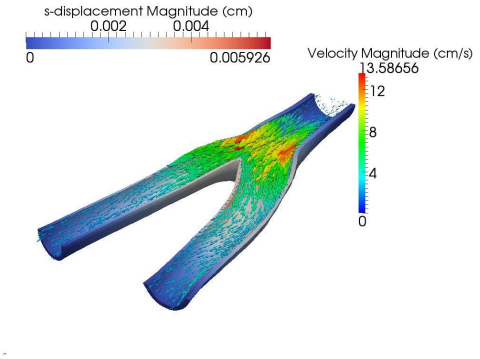
Figure 5.5: WSS[dyne/cm^2] vs. time[s]



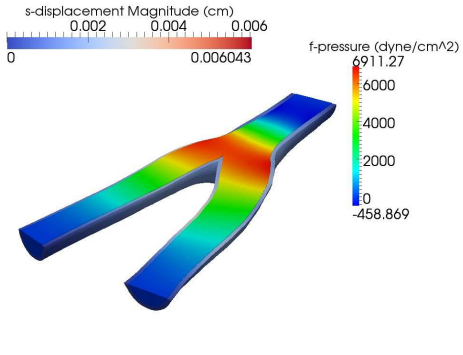
(a) F- velocity field at t = 0.0005s



(b) F- pressure and structure displacement at t = 0.0005s



(c) F- velocity field at t = 0.005s



(d) F- pressure and structure displacement at t = 0.005s

Figure 5.6: F- velocity field and pressure; Structure displacement at 2 different time-step

The fluid velocity field, fluid pressure and the structure displacement are shown in figure 5.6 at two different time-steps. Moreover, in figure 5.7, we show the pressure wave propagation at t =

0.0015s and t = 0.005s and in figure 5.8 we show the streamlines at t = 0.005s and t = 0.0055s. It is possible to see, in the last case (5.8), how the velocity field assumes a shape similar to figure 5.1.

(a) Pressure at t = 0.0015s (b) Pressure wave at t = 0.005s

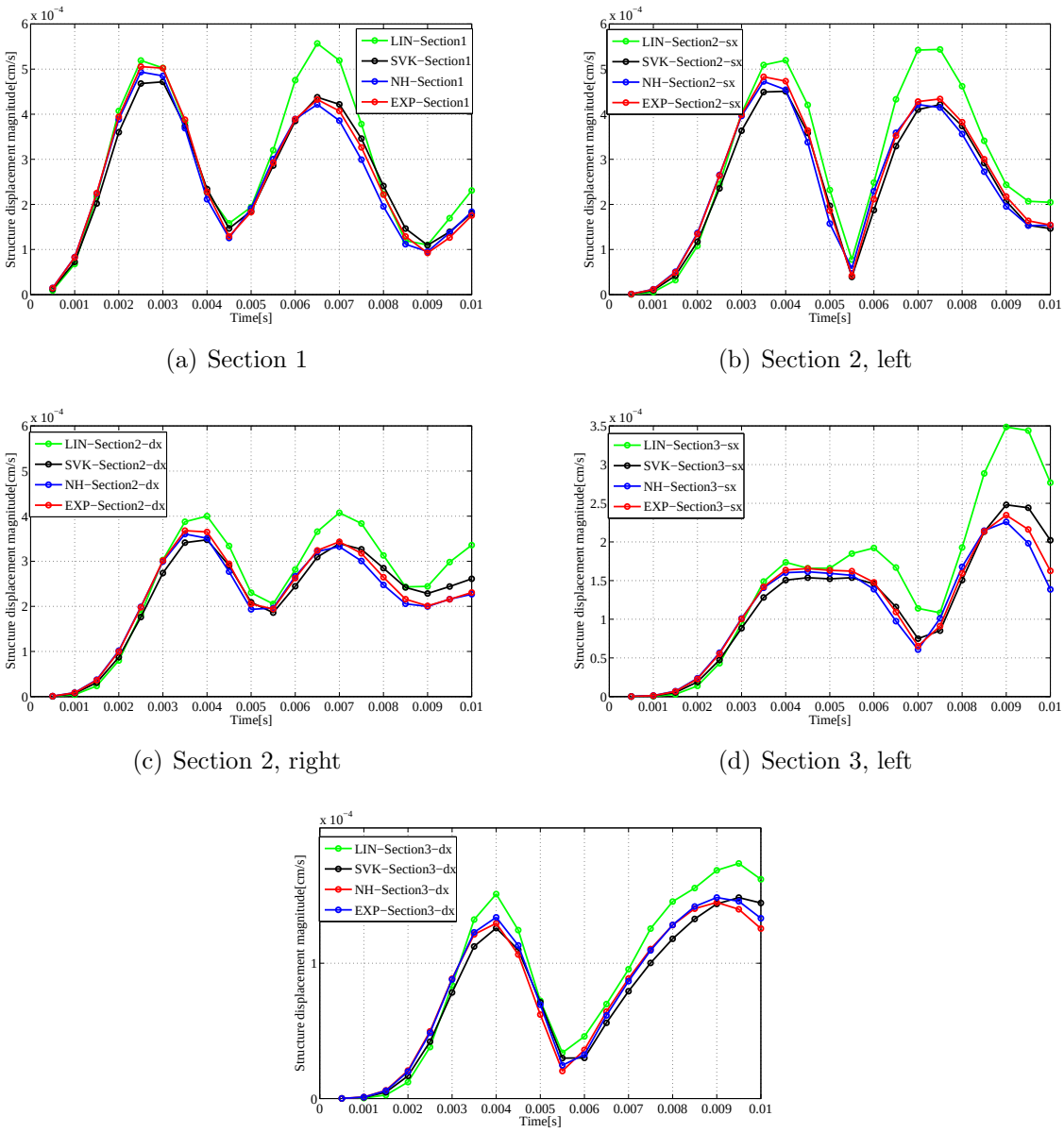
Figure 5.7: Pressure wave at two different time-steps

(a) F- streamlines at t = 0.005s (b) F- streamlines at t = 0.0055s

Figure 5.8: F- streamlines at 2 different time-step for SVK material

The mean structure displacement of the internal nodes of the 3 sections is shown in figure 5.9. It is possible to note how the nonlinear structural models cut the amplitude of the oscillations and they are more rigid than linear material.

In figure 5.10-5.11 we show the mean pressure and the mean velocity magnitude on the 3 sections presented above. Also for the fluid dynamic variables it is possible to note the same behaviour we saw for the structure displacement. In particular, about the pressure, we note the cutting of the amplitude of the oscillations using nonlinear structural models. Moreover, the pressure wave for linear material is slower than nonlinear structural models, because the wall response is less rigid as shown in 5.9. From the sequences of figures 5.10(a) to 5.10(e) it is also possible to see the space evolution of the pressure wave.



(a) Section 1

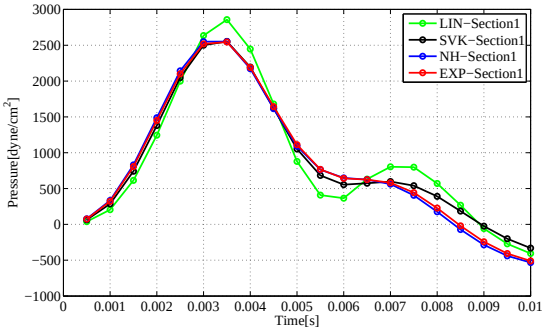
(b) Section 2, left

(c) Section 2, right

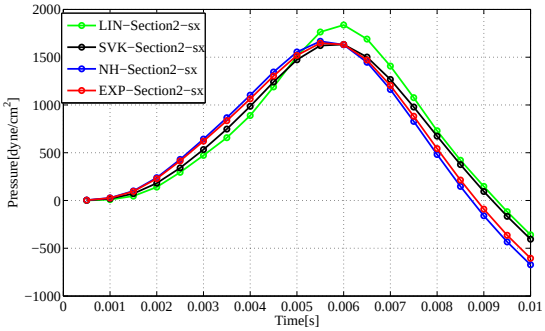
(d) Section 3, left

(e) Section 3, right

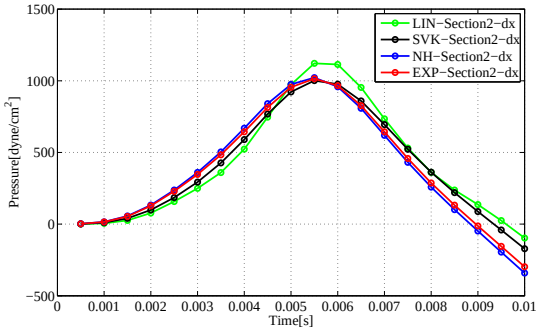
Figure 5.9: Comparison of structure displacement magnitude[cm] vs. time[s] between 3 nonlinear structural models and linear elasticity for FSI carotid simulations



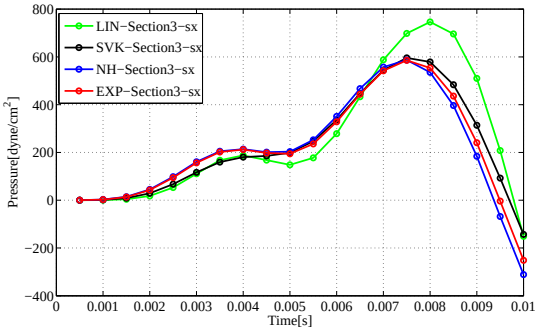
(a) Section 1



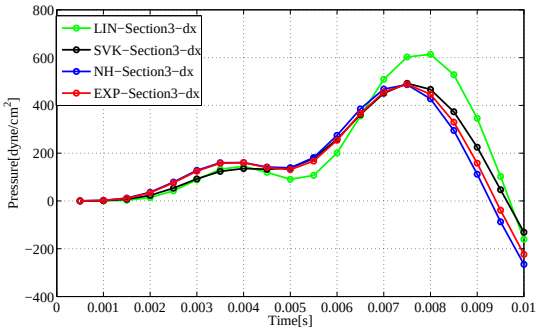
(b) Section 2, left



(c) Section 2, right

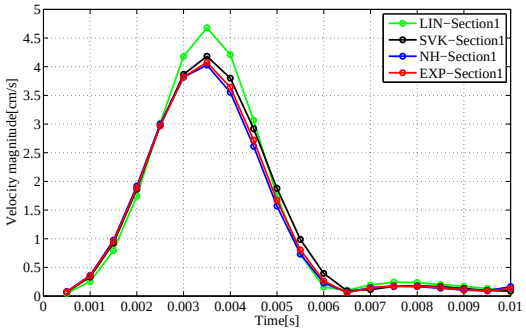


(d) Section 3, left

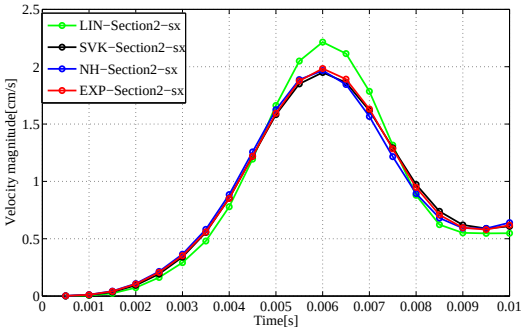


(e) Section 3, right

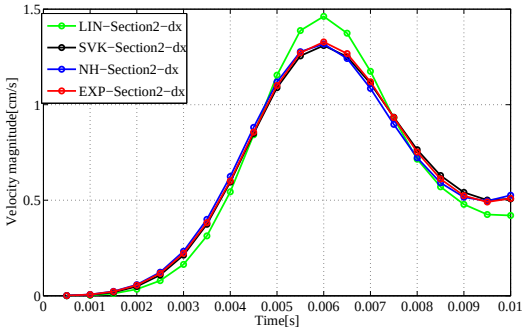
Figure 5.10: Comparison of pressure[dyne/cm^2] vs. time[s] between 3 nonlinear structural models and linear elasticity for FSI carotid simulations



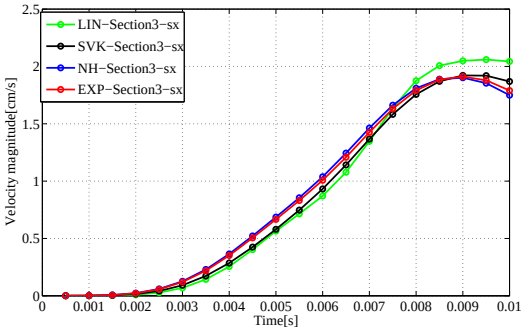
(a) Section 1



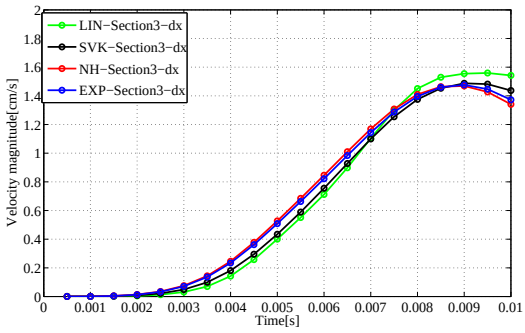
(b) Section 2, left



(c) Section 2, right



(d) Section 3, left



(e) Section 3, right

Figure 5.11: Comparison of velocity[cm/s] vs. time[s] between 3 nonlinear structural models and linear elasticity for FSI carotid simulations

Chapter 6

Conclusions and perspectives

In this Chapter, we analyze critically the results obtained in this work and described the possible perspectives.

In the second Chapter we have introduced the structural problem. In particular we have shown the nonlinear structural models implemented and tipically used for biological tissues. Moreover, we have presented the numerical test on a cube under uniaxial constant stretch for the validation of the structural solver which showed the correctness of the code. We have also set a second test case, the hollow cylinder inflation test, that shows a reasonable response from a qualitative point of view. A quantitative comparison is hard as for this test because the exact solution is not known. However, the comparison of the response of the 3 nonlinear structural models with the linear model for the hollow cylinder inflation test, reflects a consistent behaviour of the materials with respect to the test on the cube. With regard to the implementation of the structural solver, we have shown, in Chapter 4, its performances and the parallelization of the code with a scalability test on Lagrange cluster at CILEA. The results show a correct scalability of the algorithm but the performances are not optimized. In particular the NH and Exp materials show the highest dimensionless CPU-time to perfom a standard simulation. This issue is due to the construction of the Jacobian matrix that is still not optimized.

In the third Chapter we have introduced the FSI problem and we have shown the integration of the nonlinear structural solver into a FSI solver. In particular, we have used an already implemented solver based on an interface Newton Krylov method to solve the FSI problem. We have integrated our nonlinear structural solver into that algorithm and assessed its correctness on an academic test case on a straight cylinder. For this test case, we have compared the response of the linear structural model and of the nonlinear models. The results obtained show a more rigid behaviour of the wall for the nonlinear materials with respect to the linear elasticity. In Chapter 4 we have evaluated the performances of the FSI solver. The CPU-time required to perform a simulation are very high, using nonlinear structural models due to the not optimized structural algorithm as shown previously.

In the fifth Chapter we have set a haemodynamic test case to evaluate the robustness of the algorithm on complex problems. As mentioned in the Chapter 1, one of the goals of this thesis is to use the code developed in an haemodynamic contest. The results are also in this case, reasonable if compared with linear elasticity. Moreover, it is believed that the models of nonlinear structure

better describe the behavior of the arterial wall.

Overall, we have built a parallel nonlinear structural solver that is the first target of this thesis as mentioned in the first Chapter and we have valiated it on academic test cases.
We have integrated it in a parallel FSI solver and tested on both academic and haemodynamic test cases.

Future developments

As mentioned in Chapter 1, NH and Exp materials can be reused for damage models of the wall of arteries such as multi-mechanism model proposed by Robertson et al. [10]. Moreover, the nonlinear structural solver developed in this work can be easily extended to include also the anisotropic behaviour, and the pre-stress, tipically present in the arterial walls.
One of the most interesting aspects, from a hemodynamic point of view, is to accurately simulate physiological flows (for example into carotids but also in other vessels) in order to provide therapeutic indications to the doctors. For example, when a vascular surgeon operate a patient to remove an atherosclerotic plaque from the carotids, one of the critical points is to decide if operate with direct suture or by applying a patch. In the first case the operation may take up to 10 minutes with a reduction of the risks for the patient, while in the second case the operation may take up to 30 minutes, increasing the risks for the patient. The surgeon has not precise guidelines on what path to follow. So it would be interesting to provide a standard in this issue using the FSI simulations to predict the principal haemodynamic indices pre and post-intervent.

From an implementative point of view, the code is parallel but is not optimized, hence, the computational cost required can be improved.

Appendix A

Functional spaces

In this appendix we recall the main functional spaces used in this work. In particular, to derive the weak formulation of the structural and FSI problems we have used the space L^p and H^q. They belong to the family of Banach spaces where the latter is defined as follows:

Definition A.1. A *Banach space* is a normed linear space that is a complete metric space with respect to the metric derived from its norm.

For example, the space $\mathbb{R}^n$ or $\mathbb{C}^n$ equipped with the p-norm defined as:

$$||(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_n)||_p = (|\mathrm{x}_1|^p + |\mathrm{x}_2|^p + ... + |\mathrm{x}_n|^p)^{(1/p)}\,,$$

for $p < \infty$, and

$$||(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_n)||_\infty = \max\{|\mathrm{x}_1|, |\mathrm{x}_2|, ..., |\mathrm{x}_n|\}\,, \qquad \text{for } \mathrm{p} = \infty\,,$$

is a finite-dimensional Banach space.

Definition A.2. The space $\mathrm{L}^p(\Omega)$, $1 \leq p < \infty$ is the set of measurable functions $\mathrm{v}(\mathbf{x})$ in $\Omega \subset \mathbb{R}^n$ defined as:

$$\mathrm{L}^p(\Omega) = \{\mathrm{v} : \Omega \mapsto \mathbb{R} \text{ such that } \int_\Omega |\mathrm{v}(\mathbf{x})|^p \,\mathrm{d}\Omega < \infty\,,\}$$

with the following norm:

$$||\mathrm{v}||_{\mathrm{L}^p(\Omega)} = \left(\int_\Omega |\mathrm{v}(\mathbf{x})|^p \,\mathrm{d}\Omega\right)^{1/p}$$

More precisely L^p is the space of *equivalence classes* of measurable functions where the equivalence relation is defined in the following manner: v is equivalent to w if and only if v and w are equal almost everywhere in Ω [32].
A special case of the spaces $\mathrm{L}^p(\Omega)$ is that of square integrable functions $\mathrm{L}^p(\Omega)$, with $p = 2$:

$$\mathrm{L}^2(\Omega) = \{\mathrm{f} : \Omega \mapsto \mathbb{R} \text{ such that } \int_\Omega |\mathrm{f}(\mathbf{x})|^2 \,\mathrm{d}\Omega < +\infty\,,\}$$

The $\mathrm{L}^2(\Omega)$ norm is associated to the scalar product[32]:

$$||\mathrm{f}||_{\mathrm{L}^2(\Omega)} = \sqrt{(\mathrm{f}, \mathrm{g})_{L^2(\Omega)}}\,,$$

where

$$(\mathrm{f},\mathrm{g})_{L^2(\Omega)} = \int_\Omega \mathrm{f}(\mathbf{x})\mathrm{g}(\mathbf{x})\mathrm{d}\Omega\,, \tag{A.1}$$

is the scalar product in $\mathrm{L}^2(\Omega)$. It can be shown that the functions belonging to $\mathrm{L}^2(\Omega)$ are special distributions. However, it is not granted that their distributional derivatives are still functions of $\mathrm{L}^2(\Omega)$. Therefore it is appropriate to introduce the following spaces:

Definition A.3. Given $\Omega \subset \mathbb{R}^n$, the Sobolev space of order k in Ω, is the space formed by all functions of $\mathrm{L}^2(\Omega)$ having all distributional derivatives up to order k, belonging to $\mathrm{L}^2(\Omega)$:

$$\mathrm{H}^k(\Omega) = \{\mathrm{f} \in \mathrm{L}^2(\Omega) : \mathrm{D}^\alpha \mathrm{f} \in \mathrm{L}^2(\Omega),\ \forall \alpha : |\alpha| \leq k\}\,.$$

For the Sobolev spaces it is possible to demonstrate the following result:

Property A.1. *If Ω is an open subset of $\mathbb{R}^n$ with sufficiently smooth edge,then:*

$$\mathrm{H}^k(\Omega) \subset \mathrm{C}^m(\overline{\Omega}) \qquad \text{if } k > m + \tfrac{n}{2}\,.$$

A particular case of $\mathrm{H}^k(\Omega)$ is the space $\mathrm{H}_0^1(\Omega)$; this space is very useful and it is defined as:

$$\mathrm{H}_0^1(\Omega) = \{\mathrm{f} \in \mathrm{L}^2(\Omega) : \mathrm{D}^1\mathrm{f} \in \mathrm{L}^2(\Omega) \quad \text{and} \quad \mathrm{f}(\partial\Omega) = 0\}\,.$$

Nomenclature

$(\cdot,\cdot)$	Inner product in L^2	-
β	Generic elastic body	-
$\boldsymbol{\eta}_\Sigma$	Interface displacement	-
$\check{\mathbf{x}}$	Cartesian versor in x direction	-
$\check{\mathbf{y}}$	Cartesian versor in y direction	-
$\check{\mathbf{z}}$	Cartesian versor in z direction	-
$\delta\mathbf{v}$	Generic test function	-
$\delta\mathbf{x}$	Length vector	[cm]
δt	Time-step	[s]
$\mathbb{E}$	Eucledian space	-
$\mathbb{P}_r$	Polynomials' space of degree r	-
$\mathbf{n}_\Sigma$	Normal vector to interface between fluid and structure	-
$\mathbf{w}$	Interface velocity	[cm/s]
ϕ_j	Basis functions	-
H^q	Sobolev spaces	-
K	Generic element of the mesh	-
L^p	Banach spaces	-
θ	Second Newmark parameter	-
$\widehat{\tau}_\mathrm{h}$	Mesh discretization	-
$\widehat{\mathrm{V}}$	Control volume	[cm^3]
ζ	First Newmark parameter	-

Configurations and coordinate systems

$\mathbf{x}$	Coordinates in current configuration	[cm]
Ω	Current configuration	[cm^3]
Σ	Interface between fluid and structure	-
$\widehat{\Gamma}_D$	Boundary of $\widehat{\Omega}$ where it is applied Dirichlet condition	-
$\widehat{\Gamma}_N$	Boundary of $\widehat{\Omega}$ where it is applied Neumann condition	-
$\widehat{\mathbf{x}}$	Coordinates in reference configuration	[cm]
$\widehat{\Omega}$	Reference configuration	[cm^3]
$\widehat{\partial\Omega}$	Boundary of $\widehat{\Omega}$	[cm^2]

Notation

$(\cdot)_\Sigma$	Interface quantities	-
$(\cdot)_\mathrm{f}$	Fluid quantities	-
$(\cdot)_\mathrm{s}$	Structure quantities	-
$\boldsymbol{\nabla} = \boldsymbol{\nabla}_{\mathbf{x}}$	*Del* operator in current configuration	[1/cm]
$\widehat{\boldsymbol{\nabla}} = \boldsymbol{\nabla}_{\widehat{\mathbf{x}}}$	*Del* operator in reference configuration	[1/cm]

Maps

γ_Σ	Map from structure displacement to interface displacement	-
$\mathcal{A}$	Fluid map	-
$\mathcal{D}$	ALE solution map	-
$\mathcal{F}$	Fluid solution map	-
$\mathcal{L}$	Structure map	-
$\mathcal{S}$	Structure solution map	-
$\mathcal{T}$	Operator of the interface displacement	-

Tensors

$\mathbf{B}$	Left Cauchy-Green tensor	-
$\mathbf{C}$	Right Cauchy-Green tensor	-

$\mathbf{E}$	Green-Lagrange tensor	-
$\mathbf{F}$	Deformation gradient	-
$\mathbf{P}$	First Piola-Kirchhoff tensor	-
$\mathbf{R}$	Rotation tensor	-
$\mathbf{T}_\mathrm{f}$	Cauchy stress tensor for fluid	-
$\mathbf{T}_\mathrm{s}$	Cauchy stress tensor for structure	-
J	Determinant of $\mathbf{F}$	-
$\mathrm{I}_1(\mathbf{C})$	First invariant of $\mathbf{C}$	-
$\mathrm{I}_2(\mathbf{C})$	Second invariant of $\mathbf{C}$	-
$\mathrm{I}_3(\mathbf{C})$	Third invariant of $\mathbf{C}$	-

Structure quantities

α	Pre-exponential coefficient of Exponential material	[dyne/cm^2]
$\boldsymbol{\eta}$	Displacement vector	[cm]
$\ddot{\boldsymbol{\eta}}$	Acceleration vector	[cm/s^2]
$\dot{\boldsymbol{\eta}}$	Displacement vector	[cm/s]
γ	Exponential coefficient of the Exponential material	-
κ	Bulk modulus	[dyne/cm^2]
λ	First Lamè coefficient	-
λ_i, λ_j	Principal stretches of the deformation tensor	-
$\mathbf{f}_\mathrm{s}$	Forcing term	[dyne]
$\mathbf{z}$	Solid state perturbations	cm
μ	Second Lamè coefficient	-
ν	Poisson's ratio	
$\overline{\mathrm{W}}$	Strain energy function	[g·m/s^3]
ρ_s	Density	[g/cm^3]
E	Young modulus	[dyne/cm^2]
W_iso	Isochoric part of the strain energy function	[g·cm/s^3]

W_{vol}	Volumetric part of the strain energy function	[g·cm/s^3]
t_i, t_j	Principal stresses of the Cauchy stress tensor	[dyne/cm^2]

Fluid quantities

$\mathbf{d}$	Displacement vector	
$\mathbf{f}_f$	Forcing term	[dyne]
$\mathbf{u}$	Velocity vector	[cm/s]
$\mathbf{u}_f$	Fluid variables ($\mathbf{d}$,$\mathbf{u}$,p)	-
μ_f	Dynamic viscosity	poise
ρ_f	Density	[g/cm^3]
u_x	Velocity in x direction	[cm/s]
u_y	Velocity in y direction	[cm/s]
u_z	Velocity in z direction	[cm/s]
p	Pressure	[dyne/cm^2]

Acronyms

ALE Arbitrary Lagrangian Eulerian

DES DataElasticStructure class

DFSI dataFSI class

DN Dirichlet-Neumann condition

EJ exactJacobianBase template

EO ElemOper class

EOS ElemOperStructure class

Exp Exponential model class

FE Finite Element

FSI Fluid-Structure Interaction

FSIO FSIOperator class

FSIS FSISolver class

GMRES Generalized Minimal RESiduals

HES HarmonicExtensionSolver

NH Neo-Hookean model

NLSS NonLinearStructureSolver

NLR NonLinearRichardson template

OA Order of Accurancy

OS OseenSolver

RN Robin-Neumann condition

RR Robin-Robin condition

SVK St.Venant-Kirchhoff model

VKS VenantKirchhoffSolver

Bibliography

[1] *Encyclopedia Britannica, Inc.* http://www.britannica.com/, 2010.

[2] American Heart Association's (AHA) annual meeting. Cardiovascular healt crisis. *The Lancet*, 376, 2010.

[3] Heidenreich P.A. Trogdon J.G. et al. Forecasting the future of cardiovascular disease in the united states. a policy statement from the american heart association. *Circulation: Journal of the American Heart Association's (AHA)*, 1 March 2011, 2011.

[4] Health-EU The public health portal of the European Union. Website: http://ec.europa.eu/health-eu/health_problems/cardiovascular_diseases/.

[5] Holzapfel G.A. Gasser T.C. Ogden R.W. A new constitutive framework for arterial wall mechanics and a comparative study of material models. *Journal of Elasticity*, 61:1–48, 2000.

[6] Kalita P. Schaefer R. Mechanical models of artery walls. *Arch Comput Methods Eng*, 15:1–36, 2008.

[7] Holzapfel G.A. Gasser T.C. Stadler M. A structural model for the viscoelastic behavior of arterial walls: Continuum formulation and finite element analysis. *European Journal of Mechanics A/Solids*, 21:441–463, 2002.

[8] Holzapfel G.A. Ogden R.W. Constitutive modelling of arteries. *Proc. R. Soc. A*, 466:1551–1597, 2010.

[9] De Luca M. *Mathematical and numerical Models for Cerebral Aneurysm Wall Mechanics.* PhD thesis, Politecnico di Milano, January 2009.

[10] Wulandana R. Robertson A.M. An inelastic multi-mechanism constitutive equation for cerebral arterial tissue. *Biomechanics Model. Mech.*, 4:235–248, 2005.

[11] Li D. Robertson A.M. A structural multi-mechanism constitutive equation for cerebral arterial tissue. *International Journal of Solids and Structures*, 46:2920–2928, 2009.

[12] Li D. *Structural multi-mechanism model with anisotropic damage for cerebral arterial tissues and its finite element modelling.* PhD thesis, University of Pittsburgh, November 2009.

[13] Heil M. Hazel A.L. Boyle J. Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches. *Comput Mech*, 6, 2008.

[14] Degroote J. Bathe K.-J. Vierendeels J. Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. *Computers and structures*, 87:793–801, 2009.

[15] Causin P. Gerbeau J.F. Nobile F. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Comput. Methods Appl. Mech. Engrg.*, 194:4506–4527, 2005.

[16] Badia S. Nobile F. Vergara C. Fluid-structure partitioned procedures based on robin transmission conditions. *Journal of computational Physics*, 227:7027–7051, 2008.

[17] Badia S. Nobile F. Vergara C. Robin-robin preconditioned krylov methods for fluid-structure interaction problems. *Comput. Methods Appl. Mech. Engrg.*, 198:2768–2784, 2009.

[18] Nobile F. Vergara C. An effective fluid-structure interaction formulation for vascular dynamics by generalized robin conditions. *SIAM J. Sci. Comput.*, 30:731–763, 2008.

[19] Fernandez M.A. Moubachir M. A newton method using exact jacobians for solving fluid–structure coupling. *Computers and Structures*, 83:127–142, 2005.

[20] Tezduyar TE. Finite element methods for fluid dynamics with moving boundaries and interfaces. *Arch Comput Methods Engrg*, 8:83–130, 2001.

[21] Gerbeau J.F. Vidrascu M. A quasi newton algorithm based on a reduced model for fluid-sructure interaction problems in blood flows. *Math Model Numer Anal*, 37:631–647, 2003.

[22] Deparis S. *Numerical anlysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation*. PhD thesis, EPFL, 2004.

[23] Kuttler U. Wall W.A. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Comput Mech*, 43:61–72, 2008.

[24] Ciarlet P. G. *Mathematical Elasticity*, volume I. Elsevier, 1988.

[25] Holzapfel G.A. *Nonlinear Solid Mechanics: A continuum approach for engineering*. Wiley, 2000.

[26] LeTallec P. *Numerical Methods for Nonlinear Three dimensional Elasticity*. Elsevier, 1994.

[27] Ogden R.W. *Nonlinear elastic deformations*. Dover, 1997.

[28] Marsden J.E. Hughes T.J. *Mathematical foundations of elasticity*. Prentice-Hall, 1994.

[29] Fosdick R. and Silhavy M. Generalized baker–ericksen inequalities. *Journal of Elasticity*, 85:39–44, 2006.

[30] Hartmann S. and Neff P. Polyconvexity of generalized polynomial-type hyperelastic strain energy functions for near-incompressibility. *International Journal of Solids and Structures*, 40:2767–2791, 2003.

[31] Bonet J. Wood R.D. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 1997.

[32] Quarteroni A. *Modellistica numerica per problemi differenziali*. Springer, 2008.

[33] Hughes T.J.R. *The finite element method. Linear static and dynamic finite element analisys*. Prentice-Hall, 1987.

[34] Quarteroni A. Sacco R. Saleri F. *Matematica numerica*. Springer, 2008.

[35] de Luca M. Ambrosi D. Robertson A.M. Veneziani A. Quarteroni A. Finite element analysis for a multi-mechanism damage model of cerebral arterial tissue. *MOX-report*, 2011.

[36] Holzapfel G.A. Ogden R.W. Gasser T.C. Comparison of a multi-layer structural model for arterial walls with a fung-type model, and issues of material stability. *Journal of Biomechanical engineering*, 126:264–275, 2004.

[37] Holzapfel G.A. Determination of material models for arterial walls from uniaxial extension tests and histological structure. *Journal of theoretical biology*, 238:290–302, 2006.

[38] Formaggia L. Quarteroni A. Veneziani A. *Cardiovascular mathematics*. Springer, 2009.

[39] Nobile F. *Numerical approximation of fluid-structure interaction problems with application to haemodynamics*. PhD thesis, EPFL, 2001.

[40] Le Tallec P. Mouro J. Fluid structure interaction with large structural displacements. *Comput. Methods Appl. Mech. Engrg*, 190:3039–3067, 2001.

[41] Kuttler U. Wall W.A. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Comput. Mech.*, 43:61–72, 2008.

[42] Kuttler U. Wall W.A. et al. Coupling strategies for biomedical fluid–structure interaction problems. *Int. J. Numer. Meth. Biomed. Engng*, 26:305–321, 2009.

[43] Heil M. An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Comput. Methods Appl. Mech. Engrg.*, 193:1–23, 2004.

[44] Saad Y. Schultz M.H. Gmres: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7, 1986.

[45] Batchelor G.K. *An introduction to fluid dynamics*. Cambridge Mathematical Library, 1967.

[46] GetPot Input File and Command Line Parsing. http://getpot.sourceforge.net/.

[47] GNU operating system. www.gnu.org.

[48] The Trilinos Project. http://trilinos.sandia.gov.

[49] The LifeV project. www.lifev.org.

[50] Message Passing Interface Forum. http://www.mpi-forum.org.

[51] Quarteroni A. Valli A. *Numerical Approximation of Partial Differential Equations*. Springer, 2008.

[52] Heroux M.A. *AztecOO User Guide*, August 2007.

[53] Gee M.W. Siefert C.M. Hu J.J. Tuminaro R.S. Sala M. *ML 5.0 Smoothed Aggregation User's Guide*, May 2006.

[54] Sala M. *Amesos 2.0 Reference Guide*, September 2004.

[55] Sala M. *Amesos 2.0 Reference Guide*, September 2004.

[56] Umfpack homepage University of Florida. http://www.cise.ufl.edu/research/sparse/umfpack.

[57] Pozzoli M. Paride D. Ponzini R. *LifeV-parallel al CILEA: installazione e compilazione di lifev-parallel con diversi MPI. Analisi di scalabilità per un problema fluido in emodinamica*, September 2008.

[58] Kamiya A. Togawa T. Adaptive regulation of wall shear stress to flow change in the canine carotid artery. *American Journal of physiology*, 239:14–21, 1980.

[59] Zarins C.G. Giddens D.P. et al. Carotid bifurcation atherosclerosis: quantitative correlation of plaque localization with flow velocity profiles and wall shear stress. *Circ Res*, 53:502–514, 1983.

[60] Malek A.M. Alper S.L. Izumo S. Hemodynamic shear stress and its role in atherosclerosis. *The journal of the American medical association*, 282:2035–2042, 1999.

[61] Gnasso A. Irace C. Carallo C. et al. In vivo association between low wall shear stress and plaque in subjects with asymmetrical carotid atherosclerosis. *Stroke*, 28:993–998, 1997.

[62] Gibson C.M. Diaz L. Kandarpa K. et al. Relation of vessel wall shear stress to atherosclerosis progression in human coronary arteries. *Arterioscler Thromb Vasc Biol.*, 18:708–718, 1998.

[63] Gnasso A. Irace C. Carallo C. et al. Association between intima-media thickness and wall shear stress in common carotid arteries in healthy male subjects. *Circulation*, 94:3257–3262, 1996.

[64] Rossitti S. Svendsen P. Shear stress in cerebral arteries supplying arteriovenous malformations. *Acta Neurochir wien*, 137:138–145, 1995.

[65] Myers P.J. Willson A.J. On the finite elastostatic deformation of thin-walled spheres and cylinders. *International Journal of Solids and Structures*, 26(3):369–373, 1990.

[66] Chung D.-T. Horgan C.O. Abeyaratne R. The finite deformation of internally pressurized hollow cylinders and spheres for a class of compressible elastic materials. *International Journal of Solids and Structures*, 22(12):1557–1570, 1986.